The USENIX Association Magazine

;login:

volume 23 • number 3

June 1998

NEW REGULAR FEATURES:

Effective Perl Programming
Page 9
Java Performance
Page 40

inside:

features:

USENIX The Advanced Computing Systems Association

upcoming events

2nd USENIX Windows NT Symposium

WHEN	WHERE	WHO *program co-chairs*
August 3-5/98	Seattle, WA	**Thorsten von Eicken & Susan Owicki**

Large Installation System Administration of Windows NT Conference

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO *program co-chairs*
August 5-7/98	Seattle, WA	**Remy Evard & Ian Reddy**

3rd USENIX Workshop on Electronic Commerce

WHEN	WHERE	WHO
August 31-Sept. 3/98	Boston, MA	**Bennet Yee,** Program Chair **Dan Geer,** Public Key Infrastructure Session Coordinator

DEADLINES
Final Papers
July 21/98

6th Annual Tcl/Tk Conference

WHEN	WHERE	WHO *program co-chairs*
September 14-18/98	San Diego, CA	**Don Libes & Michael J. McLennan**

DEADLINES
Final Papers
July 28/98

1st International System Administration and Networking (SANE) Conference

Organized by NLUUG, cosponsored by USENIX and Stichting NLnet

WHEN	WHERE	WHO *program co-chairs*
November 18-20/98	Maastricht, The Netherlands	**Edwin Kremer & Jan Christiaan van Winkel**

12th Systems Administration Conference (LISA '98)

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO
December 6-11/98	Boston, MA	**Xev Gittler** & **Rob Kolstad,** Program Co-chairs **Phil Scarr** & **Pat Wilson,** IT Coordinators

DEADLINES			
Extended Abstracts	**Invited Talks Proposals**	**Notification to Authors**	**Final Papers**
June 23/98	June 23/98	July 21/98	October 16/98

NordU99 - 1st Nordic EurOpen/USENIX Conference

WHEN	WHERE
February 9-12/99	Stockholm, Sweden

3rd Symposium on Operating Systems Design and Implementation

Co-sponsored by ACM SIGOPS and IEEE TCOS

WHEN	WHERE	WHO *program co-chairs*
February 22-25/99	New Orleans, LA	**Margo Seltzer & Paul Leach**

DEADLINES		
Paper Submissions	**Notification to Authors**	**Final Papers**
July 28/98	October 13/98	January 6/99

5th Conference on Object-Oriented Technologies and Systems (COOTS)

WHEN	WHERE	WHO *program chair*
May 3-7/99	San Diego, CA	**Murthy V. Devarakonda**

DEADLINES		
Extended Abstracts	**Notification of Acceptance**	**Final Papers**
November 6/98	Dec. 16/98	March 23/99

Eighth USENIX Security Symposium

WHEN	WHERE	WHO
August 23-26, 1999	Washington, D.C.	**Win Treese,** Program Chair **Aviel Rubin,** IT Coordinator

DEADLINES		
Paper Submissions	**Notification to Authors**	**Final Papers**
March 16/99	April 21/99	July 12/99

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 7-11/99	Monterey, CA	**Avi Rubin,** Program Chair **Clem Cole & John Heidemann,** IT Coordinators

DEADLINES		
Paper Submissions	**Notification to Authors**	**Final Papers**
December 2/98	January 20/99	April 27/99

2nd Conference on Domain-Specific Languages

WHEN	WHERE	WHO *program chair*
October 3-6	Austin, TX	**Thomas Ball**

DEADLINES		
Paper Submissions	**Notification to Authors**	**Final Papers**
March 22/99	June 2/99	August 24/99

For a complete list of future USENIX events, access *http://www.usenix.org/events*.

contents

;login:

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) Volume 23, Number 3 (June 1998) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$40 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Editorial Staff

Editor:
Rob Kolstad <kolstad@usenix.org>

SAGE News and Features Editor:
Tina Darmohray <tmd@usenix.org>

Standards Report Editor:
Nick Stoughton <nick@usenix.org>

Managing Editor:
Eileen Cohen <cohen@usenix.org>

Copy Editor:
Sylvia Stein Wright

Proofreader:
Kay Keppler

Designer:
Vinje Design

Typesetter:
Festina Lente

Advertising

Linda Barnett <barnett@usenix.org>

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
WWW: <http://www.usenix.org>

The closing dates for submission to the next two issues of *;login:* are August 11, 1998 and October 6, 1998.

in this issue . . .

by Eileen Cohen
;login: managing editor
<cohen@usenix.org>

As you'll gather from Rob Kolstad's "motd" on page 88, *;login:*'s editor was feeling especially mellow at press time. When you dive into the rest of this issue, on the other hand, you'll find more than a few writers eager to pick a bone of some sort. Check out the letters to the editor for some controversy over spam countermeasures, USENIX board elections, programming languages, and even the inside front cover of *;login:*. The third and final installment of Neil Gunther's series on NT scalability completes his impressive debunking of Microsoft's claims on that score. Rik Farrow's musings take polite issue with Dan Geer's thoughts (published in the recent *;login:* special issue on security) on the security products market. And Warren Toomey's "Use the Source Again! Luke" gives a triumphant account of a controversy now resolved thanks to the efforts of some dedicated individuals.

On the nonpolemical side of things, we're excited to introduce two new regular series in this issue. First, Perl guru Joseph Hall provides the initial installment of a guide to effective Perl programming. In the features section, Glen McCluskey, who contributed the "Using C++ As a Better C" series over the last year, has moved from compiled to interpreted languages to focus on Java performance. Glen's will be a companion series on to our ongoing column on using Java, written this time by Prithvi Rao, who presents a neat discussion of the interoperability of Java Beans and ActiveX.

End users get more than a passing nod in this issue. Jon Howell's feature article presents Faq-O-Matic, an accessible tool for collaborative FAQ building. Dave Taylor's Webmaster column shows how to build useful error pages to help users find what they're looking for on a Web site. In the SAGE section, while Kim Trudel ponders the pitfalls of setting up a help desk, Mark Mellis codifies a working escalation tool for user support. (Kim, meet Mark.)

See you in New Orleans!

letters to the editor

What Potential Abuses?

Editor,

In the April *;login:*, you published a generally excellent article on technical "spam" countermeasures. On page 41, a thinly veiled reference was made to the MAPS RBL (see <http://maps.vix.com/>), and I would like to set the record straight on several topics.

At the last LISA, 650 people crammed into the SPAM BOF room and listened to various people discuss spam countermeasures. SunWorld Online published two articles about this event – see <http://www.sun.com/sunworldonline/>. Christenson, Farmer, and I were the principal speakers that night.

Christenson and Farmer wrote: "potential abuses have made this a practice we cannot support" and later "this solution, as currently implemented, bestows a great deal of power to an individual, so a potential for abuse is there – even though we don't suspect that any unethical activity is likely, the mere possibility of this is distressing."

Which potential abuses? If either of these gentlemen has a specific concern, I will address it. The above text constitutes some Fear, a little Uncertainty, and a bit of Doubt. But *what potential abuses?*

Christensen and Farmer continue, "In addition, the misconduct of these individuals can make the spammers appear to be victims, rather than the network abusers that we believe they are." *So what?* Sanford Wallace claims to be a victim of censorship, but *does anyone believe him?*

They finish their section on the MAPS RBL with these harsh words: "We do not support these sorts of activities in any way, shape, or form; [we] implore the employers of these methods to desist and call for other legitimate organizations to decry these methods as well."

As it turns out, I had to blackhole Earthlink's (Christenson's and Farmer's) network during several months last year when they were taking no externally visible action against spam which came from or through their network. But, while I am heartened that their time on the MAPS RBL has caused Earthlink to learn to behave as more responsible network citizens, I am shocked to see that these authors appear to be carrying a grudge about it.

Paul Vixie
<paul@vix.com>
<http://maps.vix.com/>

Christenson and Farmer respond:

We thank Mr. Vixie for his kind words about our article, but we would also like to address his comments and perhaps clarify the position we stated in the paper. He is certainly correct in stating that our section on Black Hole Routing is in reference to the MAPS RBL project.

First of all, we never had, nor have, any desire to impugn Mr. Vixie's character, and would like to offer a public apology to him in case this wasn't clear. In addition, we've changed the on-line version of the paper in an attempt to eliminate any confusion about this. While we do not support the MAPS RBL project, we certainly think that he is an honorable person who has contributed significantly to the success of the Internet.

He asks if we have specific concerns about the MAPS RBL project, and suggests that we are perpetuating fear, uncertainty, and doubt. This was not our intent, and we believe that our concerns were accurately raised in one of the articles that he cites, specifically: <http://www.eimb.rssi.ru/sunworldonline/swol-12-1997/swol-12-vixie.html>.

In this interview he directly addresses the critical problems that we see with the RBL concept. He agrees that each of the points we raise are potential problems, but has decided to continue on his original course. We are not trying to deny him his right to pursue his vision, but we also retain our right to respectfully disagree.

Finally, we'd like to refute his claim that we would slander him or his project simply because the MAPS RBL project blacklisted Earthlink. We simply find the principles behind it distasteful on philosophical grounds, and would find them so regardless of our employer or personal experiences with it. The anti-spam measures we describe in our paper were implemented not because of his project, but solely due to our (common) distaste for UCE.

more letters...

Proportional Representation?

Dear Editor:

It occurred to me as I was filling out the ballot for the Board of Directors election that we should really use proportional representation, a.k.a. single transferable vote, or "choice voting"; those of you who have voted in student government elections at MIT or UC Berkeley, or in City of Cambridge Massachusetts elections, or for the Irish Dail or the Australian Senate will know what I mean. And there is software available to do the ballot counting.

The system we use now is known to political scientists as the Block Vote; everyone gets four votes and votes for (up to) four folks at large, and the majority (or sometimes a plurality) of the membership elects the four at-large directors, leaving the rest of the membership unrepresented.

This is not fair. We have a very diverse membership, and that diversity should be reflected on the Board. (Sometimes it is, because the majority decides to elect a diverse Board. But we really shouldn't leave it to the whim of the majority.)

Other organizations sometimes elect directors by geographic region, but that just solves the problem of representing geographic diversity; it does nothing to solve the problem of representing occupational (industry vs. academe) diversity, purpose-of-the-organization diversity, etc.

There will be resistance to this idea, of course, especially from Board members who got elected under the current system and therefore see no need to change. (Of course, we have a high turnover in the Board, so maybe this is less of a problem than it is with legislatures.)

The principle behind proportional representation is very simple: majority rule with minority representation, in proportion to their respective voting strength in the electorate. With four at-large directors, a like-minded block of voters of just over twenty percent will be able to elect someone who represents them.

It will require a bylaws amendment, of course. But getting the language right will be the easy part; the Center for Voting and Democracy <http://www.igc.apc.org/cvd/> will be happy to help. The hard part will be convincing ourselves that, yes, there is a better way to elect our Board than winner-take-all.

And while we're at it, in case we ever have a contested election for an officer position with three or more candidates, we should use Instant Runoff Voting to elect our officers. (Also known as Alternative Vote or Majority Preference Voting, this system is used to elect the Australian House of Representatives and the Irish President. It's basically the degenerate case of choice voting, when you are only filling one seat.)

Steve Chessin
<Steve.Chessin@Eng.Sun.COM>

USENIX Events Calendar

Dear Editor:

Back in the misty reaches of the eighties, *;login:* published all the Association events that had been scheduled. As late as December 1996, the list of events ran from that month till June of this year. By August of last year, this had been truncated to "Upcoming USENIX Events" and spanned only 10 months. The current issue's "upcoming events" also spans 10 months. Surely, many more events have been scheduled, and better information could be given.

I was yesterday asked about a commitment for June 1999. I realized that I did not know when or where the USENIX meeting would be held (nor, indeed, whether). I also find the current "WHERE" information less than helpful. In addition to a city, a hotel, a convention center, etc., should be given. Knowing that something will be in Seattle, Boston, or San Antonio just isn't good enough.

Peter H. Salus, Bookworm

Dear Peter:

We are pleased to see that the official historian of ;login: *pays close attention to its current contents! In fact, your criticism of the "upcoming events" page indicates a failure on our part. We should point out on that page that a complete list of future USENIX events (as opposed to the "upcoming" ones, which our dictionary tells us implies approaching or forthcoming events) is available on the USENIX Website <http://www.usenix.org/events/event_calendar.html>. We note that the 2004 USENIX Annual Technical Conference will be held in Boston: May we assume you will be in attendance?*

Programming Python

To The Editor:

I am writing concerning the review of the book *Programming Python* by Terry Rooker in the April *;login:*. To someone who's already read the book, it will be clear that Rooker's piece largely succeeds in presenting a reasonably balanced picture of it as an introduction to the language. I'm more concerned about the impression that the uninitiated will get.

What I find unfortunate is how the nature of the Python language gets lost in the review. The final paragraph is particularly confusingly put in that it says, "With the exception of its use as a scripting language, almost all the claims made for Python can be made for Java, . . ." While this is certainly true, the same could be said of any number of modern programming languages. This sounds a bit to me like saying "with the exception of its ability to fly, most of the claims that can be made for the helicopter could also be made for the automobile." Yes, yes, both of them can get you from point A to point B, but Python *is* fundamentally a scripting language and Java is not, and readers really ought (if my understanding of the purpose of your book reviews is correct) to be made aware how different Python is from some of the other languages around, and given a sense of what they will be getting for their money if they buy this book. A book review is not a language tutorial, but if there's no feeling given for what you can do with a language (and a scripting language, at that!) having this degree of referential transparency, and of the expressive power that results from granting virtually all objects first-class citizenship, then something important is missing.

I also feel something should have been said about how enlightening it is to be able to experiment with the ideas involved in object-oriented programming without having to hassle with the complexity of the declaration syntax and semantics inherent in a C++ or a Java. For someone already conversant with OOP, this may not seem like much, but to a programmer who has never quite found the time to make those first few steps, Python as presented in Lutz' book could well be a real eye-opener.

John D. Garberson
<john.garberson@nexos.com>

Rooker replies:

I only have a couple of comments in response to John D. Garberson's letter. I'm glad he found my review a fair description of the book. He seems most concerned that I didn't explain the differences between Python and other alternatives, although he admits the purpose of the review is not to provide a language tutorial.

In reality I feel I slighted the book. How do you condense comments about an 800 page book into approximately 2000 words? There are many other things I would have liked to say. So yes I may not have addressed Python's special features as well as I could. I would have mentioned an online reference for readers to pursue, but I have yet to find a reasonable tutorial introduction to Python. The tutorial on the Python web site is somewhat confusing if you aren't already familiar with Python or the 3 styles of programming used by the language (procedural, functional, and object-oriented).

Mr. Garberson's other comments involve pointing out how Python allows you to

more letters...

program in an object-oriented style without all the requirements of object orientation:

"I also feel something should have been said about how enlightening it is to be able to experiment with the ideas involved in object-oriented programming without having to hassle with the complexity of the declaration syntax and semantics inherent in a C++ or a Java.."

I find this comment interesting as it reflects an attitude towards object-oriented programming that I've often found. Object-oriented programming was developed to enforce certain software engineering principles, specifically encapsulation and abstraction. Yet so many programmers work so hard to find ways around these "requirements." The "hassle" Mr. Garberson refers to are the enforcement of these principles. I once attended a vendor training course on an object-oriented language and the instructor spent one afternoon explaining how to create global variables to prevent the overhead of using object methods to access those variables! Even worse, although 75% of the class claimed to be familiar with object-oriented programming, only 2 of us (10%) realized that the pseudo-global variable defeated the whole rationale for using an object-oriented language in the first place.

What some find to be freeing them from the "hassles" of a certain programming paradigm, I find to be frightening. I mentioned that Python allows you to mix procedural, functional, and object-oriented styles of programming. For experienced programmers, knowledgeable in each paradigm, it is probably not a problem. For everyone else, there is potential for trouble. Each style makes certain assumptions and mixing the styles can invalidate these assumptions. While this mixing of styles is one of Python's main attractions, it is also a major source of concern.

Clarification

Dear Editor:

My February *;login:* article on "auditing" included a script that processed everything in the `/tmp` directory. For security reasons another "processing" directory, preferably accessable only by the auditing account and root, should be used. Using `/tmp` is problematic, in that all users have read and write access to that directory. I would like to thank Hobbit for pointing this out.

Phil Cox
<pcc@ntsinc.com>

Be a Conference Reporter or Photographer

Help *;login:* capture conferences in action! We're looking for both volunteer session summary writers and volunteer photographers for the following events:

2nd USENIX Windows NT Symposium
August 3-5, 1998
Seattle, Washington

Large Installation System Administration of Windows NT Conference
August 5-7, 1998
Seattle, Washington

3rd USENIX Workshop on Electronic Commerce
August 31-September 3, 1998
Boston, Massachusetts

6th Annual Tcl/Tk Conference
September 14-18, 1998
San Diego, California

Please contact Eileen Cohen, *;login:* Managing Editor <cohen@usenix.org>, if you're interested in contributing.

SAGE news & features

80/20 Across the Board

by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections, and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

It's April as I write this for the June *;login:* deadline. I've just finished reading my February issue. Several authors talked about their New Year's resolutions (the deadline for the February issue is in December). Reading their articles made me think about my own priorities.

I was reminded that it was this week last year, while writing my *;login:* column, that I found out about the unexpected passing of a co-worker and good friend of mine. She was only 54, not old by anyone's standards, and increasingly younger as I march through my own birthdays.

Those close to me can tell you that losing her really threw me for a loop – partially because I expected to be able to continue our friendship that I enjoyed so much and, probably, because it hardly felt "fair" that she was gone at 54. She very much still had a "list" of things she intended to do; she wasn't nearly done living. She was a co-worker I found depth in. She was great to work with on a technical basis, but what drew me closer to her was her perspective on life. She had a "hard work" ethic, but she also advocated balance in everyone's life and practiced it in her own. She always encouraged me to seek high standards in work and balance in life.

In my last article I discussed my 80/20 principle in measuring job fit. Last April I made a personal decision to take the same 80/20 rule and apply it across all my commitments. (Frankly, I have made that decision on numerous occasions, but last April I made it more earnestly, and I've stuck with it better than ever before.) It's just a simple extension of what I outlined last time; the difference is you apply the same methodology across all your time, not just 9 to 5. Remember the equation: for every project you take on, ask yourself if 80% of what it requires to do the job are things you want to be doing. If it doesn't meet that criterion, don't say "yes" to the commitment.

In my case, I made the resolution to give nonwork opportunities a more equal weight with those ever-demanding work-related tasks. Now when someone asks me to take on a new project at work, I consider how that fits into my long-term goals, both in my career and in my life, and how it would affect the way I spend my collective time. Once you get the hang of it, the 80/20 question is easy to remember to ask yourself, and it helps you make decisions that balance your time in a way that you'll be happy with in the long run.

The hard part, of course, is sticking to it. I'm convinced it's at least as difficult as any ill-fated diet! If you're like me, your interests far outweigh your time and resources. It's almost dangerous for me to read the newspaper; I always learn about a worthy cause or an exotic destination. It seems I'm in perpetual-shopping mode on the aisles of life, and it looks so wonderful I want to try it all!

But life doesn't work that way, and last April was a harsh reminder of that for me. We don't get an infinite number of years to try everything we'd like to. So I decided to work a little less and live a little more, if I could.

Of course, nothing is perfect. There are fire drills in everyone's life: crises you don't plan for, opportunities you can't pass up, digressions that turn into full-fledged projects. But asking yourself the 80/20 question in all aspects of your life can help you plan your time so you don't miss out. I think it's worth a try.

Thanks To You

by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

One of the problems with having a board full of sysadmins is that we are all used to our problem-filled, interrupt-driven workdays, where we just elbow others out of the way, go in and fix what's broken, then jump on to the next problem. Teamwork is not one of our most common characteristics or requirements. When seven such people try to work together, some days are a little rougher than we'd like. Agreeing on goals, let alone working in concert to achieve them, is difficult.

For the last couple of years the USENIX Board has tolerated our growing pains on the SAGE Board by assigning one of their own as a liaison and by staying "out of harm's way." To their credit, they've given SAGE tremendous freedom and support, allowing us to accomplish things we just couldn't have done otherwise. To their even greater credit, they assigned Eric Allman as their liaison.

When we got off track, Eric gently pushed us back on. If that didn't work, he pushed harder. When we didn't seem to be working together, he stepped in and pointed it out. When necessary, he did so in strong enough terms that we would stop and reassess our approach.

When things were clicking along, Eric was along for the ride. His contributions to our discussions were always on-point, disinterested bystander observations where appropriate, USENIX Board representative when asked.

Those who know Eric are not going to be surprised at these comments, nor when I say that he was never afraid to speak his mind in, or out of, our meetings. He was also first in line to ensure that the SAGE Board had the authority and room to make our own decisions. That's a very fine line to walk, but he did so admirably.

We all know Eric Allman for his technical accomplishments. As he now steps aside from the USENIX Board, and thus the role as liaison to the SAGE Board, let me add that his organizational talents and people skills are also top notch. Without your assistance, Eric, we would not be anywhere close to as far along as we are.

Thank you, on behalf of all of SAGE.

Do Help Desks Help?

by Kim Trudel

Kim Trudel is a member of the SAGE STG Executive Committee.

<kim@mit.edu>

I find myself in an interesting situation: building a help desk *de novo*. This includes everything from staffing, to identifying the right technology, to developing work processes, to renovating office space. As work on the project begins, I think back to my experience as a client of numerous help desks and wonder whether a successful help desk exists.

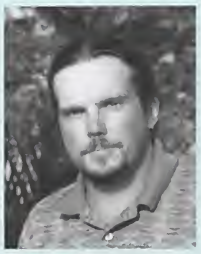
The challenges are numerous. First, how can a help desk be structured so it is efficient? Users, and I'm no exception, always seem to want immediate attention. They want to speak with someone now! As a result, they pick up the telephone, which means that someone must be there to answer the call. People costs are the single most expensive aspect of any business. A one-to-one ratio of help desk staff to callers can be extremely inefficient.

Influencing behavior so users feel as comfortable sending an email message as they do phoning could allow the help desk staff to do some multitasking, thereby increasing efficiency. Can this be done? How?

Second, how can the work be made more rewarding and less stressful? Let's face it, help desk staff, like sysadmins, tend to hear from people when there are problems. Facing endless hours of problem reports or questions that could have been easily answered by the book on the shelf is draining on even the best of us. What's the right mix of time on the help desk and time for other projects?

Many users complain about the low-quality service they get from help desks. There is incredible demand for competent help desks. Intel recently announced an answer center that will answer questions about any manufacturer's computer. Is it possible to deliver high-quality service, manage the operation efficiently, and retain staff? I'd entertain any thoughts you may have via email.

Effective Perl Programming: Without a `map` You Can Only `grep` Your Way Around



by Joseph N. Hall

Joseph N. Hall is the author of *Effective Perl Programming* (Addison-Wesley, 1998). He teaches Perl classes, consults, and plays a lot of golf in his spare time.

<joseph@5sigma.com>

Welcome to Effective Perl Programming, the column. In this and coming articles I'm going to discuss ways that you can use Perl more effectively, whether by mastering Perl idioms, using Perl modules, or finding new applications for Perl programs. I begin by covering two very powerful but underused language features: Perl's `map` operator and its cousin, the `grep` operator. I'll cover the basics quickly, then show you some useful techniques and a few neat tricks as well.

The Basics of map and grep

The `map` operator creates a transformed copy of a list by evaluating a specified expression or code block for each element in the list. Each time the expression or block is evaluated, `$_` contains the value of the current element. The result from the expression or block is appended to the list returned by `map`. The syntax for `map` looks like this:

```
@result = map expression, list
@result = map { code } list
```

Rewritten without using `map`, the effect is like this:

```
@result = ();
foreach (list) { push @result, expression; }
```

For example:

```
@times_ten = map $_ * 10, 1..10;
```

returns the list `(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)`, and

```
@uppercased = map { ucfirst $_ } qw(george jane judy elroy);
```

returns the list `('George', 'Jane', 'Judy', 'Elroy')`. The transform expression

The `map` *operator is more versatile and can do anything that* `grep` *can.*

(or block) is evaluated in a list context. It does not have to return a single element – it can return two or more or none at all (an empty list). More on that later.

The `grep` operator resembles the `map` operator syntactically:

```
@result = grep expression, list
@result = grep { code } list
```

However, unlike the `map` operator, which constructs a transformed copy of a list, the `grep` operator selects items from a list. The selection expression (or block) is evaluated for each element of its argument, with `$_` set to the current element. If the result is true (anything other than the empty string or the string `'0'`), a copy of the element is appended to the result from `grep`. For example:

```
# Returns (2, 4, 6, 8, 10).
@even = grep { not $_ % 2 } 1..10;

# Returns a list of text files in the current directory.
@text_files = grep -T, glob "*";

# Classic grep – imitating Unix grep. Prints lines containing the
# word 'Joseph'.
print grep /\bJoseph\b/, <>;
```

The `grep` operator has been around for a long time, but the `map` operator is new in Perl 5 (as much as anything that is four years old can be called "new," anyway). The `map` operator is more versatile and can do anything that `grep` can:

```
# Another way to get a list of text files.
@text_files = map { (-T) ? $_ : () } glob "*";
```

map and grep Idioms

The `map` operator is obviously useful for simple one-to-one transformations:

```
# Print out the contents of a hash.
print map "$_: $hash{$_}\n", sort keys %hash;
```

Be careful, though: this approach creates a lot of temporary structures in memory. For a very large hash it would be more appropriate to use an `each` loop:

```
while (($key, $val) = each %hash) { print "$key: $val\n" }
```

Using `map` to construct hashes is an important idiom. You can construct existence hashes that are used to test whether a particular value has been seen; in this case, set all the values in the hash to 1 (or some other "true" value). You can also use `map` to construct hashes where the value is computed from the key. To use `map` to construct a hash, return two values for each original element – the key and its corresponding value.

```
# Create keys for all the "words" in $text, so that we can test for
# a word later with if $seen{$word}.
%words_seen = map { $_ => 1 } split /\s+/, $text;

# After this, $file_size{$file} gives -s $file – saves time if
# we need to use it more than once.
%file_size = map { $_ => -s } @files;
```

The `map` operator is handy for "nesting" and "slicing" multidimensional data structures. Using an anonymous array (or hash) constructor inside `map` creates nested structures. For example, you can blend parallel arrays into a single 2-d structure:

```
# Blend @x, @y, and @z into a single 2-d array @xyz ... $xyz[0][0]
# is $x[0], $xyz[0][1] is $y[0], and so on.
@xyz = map [$x[$_], $y[$_], $z[$_]], 0..$#x;
```

You can use the same technique to create a hash of arrays:

```
# Cache the results from stat into a hash of arrays ... then
# $info{'file'}[7] gives the size of 'file', $info{'file'}[5]
# gives the owner's uid, and so on.
%info = map { $_, [ stat $_ ] } @files;
```

Extracting a slice of a nested structure is just as easy. Just use a subscript inside map:

```
# This will extract @x from @xyz (undoing what we did above) ...
# $x[0] is $xyz[0][0], $x[1] is $xyz[1][0], and so on.
@x = map $_->[0], @xyz;
```

The grep operator isn't as versatile as map, but it is usually the most succinct way to select items from a list. Don't forget that it can be used on complex structures:

```
# Select elements from @xyz whose "coordinates" are all >0.
# @gt_zero is still a 2-d array with the same organization as @xyz.
@gt_zero = grep {$_->[0] > 0 and $_->[1] > 0 and $_->[2] > 0} @xyz;
```

Cool Tricks with map

You can use map to read several lines of input at a time:

```
# Read 10 lines from STDIN.
@ten_lines = map scalar(<STDIN>), 1..10;
```

The "Schwartzian Transform" (named after fellow Perl trainer and author Randal L. Schwartz) is a sort surrounded by maps. It is generally preferred over other techniques when the sorting process requires time-intensive key transformations:

```
# Sort files in descending order of size.

@files_by_size =
map { $_->[0] }              # 3. slice out the original list, now sorted
sort { $b->[1] <=> $a->[1] } # 2. sort the list of tuples
map { [$_, -s $_] }          # 1. create a list of tuples by nesting
@files;                      # the data to be sorted
```

You can use map for some set operations. Here is an example of using it to find the elements in one hash (%hash1) that are not in another hash (%hash2). Depending on the relative sizes of the hashes involved, this can be more efficient than other methods (like using the delete operator):

```
# keys %result contains 2 4 6 7 8 9 when this is done.

%hash1 = map { $_, 1 } 1..9;     # some sample data
%hash2 = map { $_, 1 } 1, 3, 5; # more sample data
%result = map { $_, $hash1{$_} } grep { not exists $hash2{$_} }
      keys %hash1;

# Another way to do the same thing, with delete.

%result = %hash1;
delete @result{keys %hash2};
```

Because map's transform expression is evaluated in a list context, using map in combination with a pattern match that contains some parentheses can produce unusually succinct code:

```
# Create a hash of user name vs. user id from lines in /etc/passwd.

open PASSWD, "/etc/passwd" or
  die "couldn't open password file: $!\n";
%name_to_id = map /(.*?):.*?:(.*?):/, <PASSWD>;
```

The map operator can even be useful for some string operations:

```
# Convert a string like 'ABC' into its
```

The "Schwartzian Transform" is a sort *surrounded by* maps.

```
# hex equivalent, '\x41\x42\x43'.
$hexed = join '', map { sprintf "\\x%x", ord $_ } split //, $str;
# An alternative using s///, which is slightly slower
# for long strings.
($hexed = $str) =~ s/(.)/sprintf "\\x%x", ord $1/ge;
```

That should be enough for now. I hope you've enjoyed this little tour of `map` and `grep`. My next column will be something of a change of pace – I will introduce object-oriented programming in Perl.

by Ping Huang

Although he knows very little about 3D graphics, Ping Huang is a software engineer at Silicon Graphics. Previously he worked on the Gauntlet firewall product and on the IRIX kernel; currently he is working on system software for the new line of Intel/NT-based graphics workstations.

<pshuang@alum.mit.edu>

Some Emerging Peripherals Interconnect Technologies: USB and 1394/Firewire

Much press has been made about the increasing push of Wintel systems into what formerly has been nearly exclusively the domain of UNIX workstations and servers. Closer to home, substantial ink has been used to discuss specifically how to administer such heterogeneous environments. However, even if you administer an environment that continues to consist entirely or nearly entirely of UNIX workstations and servers, you should keep an eye on developments in the PC world as UNIX system vendors increasingly incorporate commodity I/O technology from the Wintel world into their systems. From the customer's point of view, the use of components and peripherals that benefit from the economies of scale of the Wintel world can be a great thing, especially in arenas where alternative technologies that the UNIX system vendors could incorporate into their systems instead don't offer any compelling value add.

Examples of I/O technology that have already been widely adopted among UNIX system vendors include PS/2 ports for keyboards and mice and VGA-style connectors for monitors. There's little meaningful value add in having keyboards and mice that use proprietary connectors, and system vendors that continue to use them make it difficult for customers with ergonomic needs or unusual requirements to obtain what they need. Similarly, few tears should be shed now that the drawer full of oddball monitor cables can largely be packed away and forgotten.

Examples of less widely adopted technologies include IDE/ATAPI and PCI. The competitor for IDE/ATAPI, SCSI, offers higher throughput, especially under multitasking operating systems, as well as the flexibility to attach many peripherals to a single SCSI bus. These factors help to justify SCSI's generally higher cost. As for PCI, only recent revisions of the PCI standard provide the level of bandwidth that workstations' proprietary expansion buses have had for years. Furthermore, because PCI add-on cards require their own individual specific device drivers, and UNIX vendors have generally only supplied device drivers for a limited number of cards, customers don't benefit much from the existence of the thousands of low-cost mass-market PCI cards.

They may be able to plug such cards into their UNIX workstations, but cannot make use of them.

Several relatively new peripherals -interconnect technologies are emerging in the Wintel world – Universal Serial Bus (USB) and the 1394 serial bus (also known as Firewire, a name trademarked by Apple). USB and 1394 have many similarities, but are not identical and should not be confused with each other. As is evident from their names, these technologies are both serial buses rather than parallel buses, which allows for smaller connectors and thinner cables, holding costs down. They both support supplying a modest amount of electrical power to peripherals, a boon for user convenience and a bust for manufacturers of power strips, and allow daisy-chaining dozens of peripherals together. They both support hot-plugging peripherals and specify that identification information be embedded within peripherals so that the host system can automatically identify a device (including one that was just hot-plugged) and load the necessary software support for it. The host can also identify the resource requirements of the peripherals' device drivers and configure hardware and software to avoid any conflicts over interrupts, and such, without requiring user intervention (thus avoiding the unpleasant experience of having to set DIP switches on old ISA boards or having to set SCSI IDs, perhaps through obscure jumper settings). These bus characteristics may be able to elevate Plug'N'Pray (a sardonic description of what Microsoft Windows 95 tries to offer) to true plug and play.

Both buses were also originally largely intended for peripherals that go outside of the system chassis. However, Device Bay, a technology initiative supported by Compaq, Intel, and Microsoft, would provide externally accessible bays on desktop and laptop systems that would accept "cartridges" that are, roughly speaking, the size of a slim CD-ROM drive. Device Bay peripherals would be supplied both USB and 1394 ports, as well as copious amounts of power, because one of the primary target markets for Device Bay would be mass storage devices, many of which require more power than can be delivered over either of the serial buses. In theory, Device Bay could drive enormous penetration of USB and 1394 support by both peripherals and systems vendors, but in practice, there appear to be numerous snags to its wide adoption by system vendors; many of these snags are not entirely technical in nature.

There are also many significant differences between USB and 1394, of course. USB is considerably more mature technology than 1394. More peripherals are available for USB, and Intel chipsets used at the heart of a percentage of PCs shipping today support USB. By contrast, I believe there are today no system motherboards that include integrated 1394 support; one must purchase a separate 1394 PCI controller card. The silicon necessary to support USB is extremely inexpensive, a necessity because some of the peripherals that fall into USB's target markets may retail for less than $10. By contrast, although one of the design goals of 1394 was to keep costs as low as possible, 1394 was not targeted at quite that low a cost level. USB passes the Fry's Electronics test – I was able to go down to several branches of this infamous Silicon Valley computer and electronics store and ask employees for help finding a USB cable and receive actual help, rather than a blank stare, in return. 1394 sadly fails this test today. Operating system support also varies. USB is partially supported by the more recent Microsoft Windows 95 OEM preinstalled releases; full USB and 1394 support are promised for Windows 98 and Windows NT 5.0 and can be previewed in the respective beta releases of the two OSs. 1394 is fully supported today by Apple's MacOS 8.0, which isn't surprising because Apple had a considerable head start in that it originally developed the underlying technology under the name Firewire before deciding to promulgate it as an industry standard.

There are also many significant differences between USB and 1394, of course. USB is considerably more mature technology than 1394.

Vendors of non-Wintel systems will be forced to make difficult decisions about how much support for USB and 1394 will need to be incorporated into their hardware and software.

Other differences are more specific to technical variations between the two. USB has a maximum data transfer rate of 12 megabits per second (1.5 megabytes per second), and therefore potential target peripherals must be low- to medium-speed devices, such as human input devices, modems, moderate fidelity audio, or digital camera still imaging. But even a 12x CD-ROM drive, not considered particularly speedy today, would exceed the transfer rate that USB can handle. By contrast, 1394 data transfer rates are much higher. 1394 has been ratified as an IEEE standard – IEEE 1394-1995 – which defines a top speed of 400 megabits (50 megabytes) per second. (Work continues on a future standard that would extend the top speed upward into the gigabits per second range.) 1394's current top speed of 400 megabits per second falls short of Ultra2 SCSI, but otherwise exceeds the bandwidth of any other mass storage interconnect technology that isn't effectively restricted to high-performance servers due to cost considerations (e.g., Fibre Channel). USB supports bidirectional data communication between the host and the peripheral, but the host is considered a special node on the USB bus. 1394, however, fully supports peer-to-peer data transfers, allowing a host system to set up data transfer between two peripheral devices, such as a digital camera and a digital video recorder, and subsequently not have to be burdened by being involved with the actual video data transfer. In addition to its higher bandwidth making digital video (not just still imaging) possible, 1394 also contains explicit low-level support in its bus protocol for priority delivery of time-critical data, which it calls isochronous, as opposed to the normal asynchronous data transfers. Its ability to reserve bandwidth for isochronous data makes 1394 more attractive than USB for more sophisticated and professional-caliber multimedia applications. (USB also supports isochronous data delivery, but targets consumer-level multimedia and telephony applications).

Only time will tell for certain whether these two technologies will succeed to such an extent that vendors of non-Wintel systems will incorporate hardware and software support into their systems. USB's greater maturity, less ambitious capabilities, and lower cost increase its chance of becoming ubiquitous. 1394's future prospects are less clear; its potential is greater, but so are its complexity and cost. To quote from the Intel presentation at a recent 1394 Trade Association meeting, 1394 faces the danger of becoming a Swiss Army knife that can in theory do everything but can do nothing well. If that happens, 1394 will fail to take off.

Vendors of non-Wintel systems will be forced to make difficult decisions about how much support for USB and 1394 will need to be incorporated into their hardware and software. In addition to system software support for the buses themselves, different peripherals will call for individual device drivers, as is the case with PCI. Unlike with PCI, there appears to be at least some effort by peripherals vendors to establish standards for how their peripherals interface to software such that perhaps a single device driver will be able to control multiple peripherals that fall into the same class of functionality but are from different peripherals vendors. Any such standardization will be to the benefit of non-Wintel vendors.

World Wide Web sites promoting these technologies:
<http://www.usb.org/>
<http://www.1394ta.org/>
<http://www.device-bay.org/>

Riding the Escalator

by Mark K. Mellis

Mark K. Mellis is principal engineer of Mellis and Associates, a Silicon Valley consulting firm specializing in Unix systems and IP networking. His interests include computer telephony, reliable network infrastructure design, jazz, and rubber chickens.

<mkm@mellis.com>

I've always been challenged by the management of flipped-out users – ones who have a problem that, to them, is the most pressing thing in the world. I "feel their pain," yet I have to balance their needs in the context of the entire organization that I support. As usual, I'm not the only one with this problem, and also as usual, the other folks have a pretty good tool for managing the problem. In my best "egoless sysadmin" style, I've adopted it for my own use. My new magic bullet is the defined escalation procedure.

How It Works

You classify your incoming calls (or tickets or issues or whatever your favorite euphemism might be) by priority. Each classification has expected service goals associated with it. Those service goals are published. If a call isn't progressing according to the service goals for its classification, it moves to a higher classification, and those involved are notified. This continues until the call is closed or it reaches the top of the escalation ladder. The goal is never to have to escalate calls beyond their initial level. In addition to autoescalation, a call can be escalated at will by the submitter or by the sysadmin organization. However, the path for each call is defined, and all parties know in advance what their options are when progress isn't as apparent as they wished it was. By defining the path through which a problem is addressed and providing a clear mechanism to escalate the issue when necessary, you help move frustration away from the personalities involved and focus on the problem and its resolution.

Implications for the Organization

A defined and published escalation procedure helps relieve users' stress because it gives them a way to bring their problems to higher authority when they don't think they are getting the attention they deserve.

Implicit within the escalation procedure is a defined priority for each issue. When you have a clear idea of the priority of a problem, you can allocate resources to it in a fair manner, and if it is escalated, the increased resources of the higher level become available, assuring that appropriate resources are channelled to critical problems.

Managers don't like to be blindsided. There's nothing like having your boss called on the carpet for a problem you haven't told her about yet. If escalations are actively managed, these situations are less likely to occur.

Don't forget, you must define the escalation procedure in writing and publish it throughout your user community and their management. If users don't know about it, they can't use it.

Like all policies and procedures, defined escalation is unlikely to succeed as a sysadmin tool without management support. Your boss needs to "sign up" to be on the escalation path and needs to back you up in a constructive manner when contacted by a constituent who is escalating an issue.

Defined escalation is not a project management tool. Project work should be managed separately.

Here is an example of the escalation procedure for a medium-sized company:

Routine request

Submit via Web form, email, or telephone call to x4357 (HELP), or call: Duty Sysadmin pager +1 408 555 5554.

Worked by duty sysadmin.

Routine requests are account creations, file restores, network moves, alias maintenance, and so forth.

Status reported at time of request and every 16 business hours thereafter.

Escalates to next level in 24 business hours.

Minor outage, impairment

Submit via Web form, email, or telephone call to x4357 (HELP), or call: Duty Sysadmin pager +1 408 555 5554, Duty manager pager +1 408 555 5555.

Worked by duty sysadmin, duty sysadmin manager, submitter's manager automatically notified.

Minor outages affect eight or fewer users. For example, a single 10BaseT hub failure or a single workstation failure is a minor outage. Impairments are error conditions that have not yet caused an outage. For example, high error rates on a disk drive or network connection are impairments.

Status reported at time of request and every 4 business hours thereafter.

Escalates to next level in 8 business hours.

Major outage

Submit via Web form, email, or telephone call to x4357 (HELP), or call: Duty Sysadmin pager +1 408 555 5554, Duty Manager pager +1 408 555 5555, IS Director pager +1 408 555 5556.

Worked by duty sysadmin and other resources as dispatched by management. Duty Sysadmin manager, IS Director, submitter's manager, and director automatically notified.

Major outages affect nine or more users or a major service, such as DNS or Internet connectivity. A major fileserver failure or a security incident in progress is a major outage.

Status reported at time of request and hourly thereafter.

Escalates to next level in 4 business hours.

Disaster

Submit via Web form, email, or telephone call to x4357 (HELP), or call: Duty Sysadmin pager +1 408 555 5554, Duty Manager pager +1 408 555 5555, IS Director pager +1 408 555 5556, Office of the President pager +1 408 555 5557.

Worked by all available resources. All Sysadmin management, all directors, and the Office of the President automatically notified.

Disaster is a company-wide failure of IS infrastructure. A fire in the data center or a major security penetration is a disaster.

Status reported at time of request and hourly thereafter.

This is the highest escalation level.

Let's walk through a few cases. The network drop for Bill's workstation fails. He calls x4357 and opens a routine ticket. The ticket number he gets in return is his initial notification. The duty sysadmin comes by in 20 minutes, repatches him to a working port, and closes the ticket. Work flow is normal, so there is no escalation.

Susan needs an alias created to support her latest project and wants it done immediately. She knows that alias creation is a routine event and can reasonably take up to three business days. She can escalate it, but her management will be automatically notified if she does. Susan chooses to wait. The alias is created in a timely manner and the ticket is closed.

Jake needs a special CAD package updated on his workstation. He opens a routine ticket on Monday, but his request gets lost in the workload. On Thursday, it is autoescalated from routine request to minor outage, and the appropriate managers are notified. Jake's software is updated early Friday morning by a chastened sysadmin.

Dorothy requests that her workgroup be moved to its own subnet to improve performance. Router interfaces are in short supply and the request can't be fulfilled in the designated time, so the sysadmin escalates the request. During the subsequent automatic management review, Dorothy is persuaded that she didn't need her own router interface after all.

Defined escalation is a tool, and will benefit you only if you use it properly. It must be accepted by your organization, including your user community. You have to be willing to live with its consequences. In return, you may reap the benefits of more businesslike interactions with your constituency and emergencies that are really "emergent."

Request for Proposals for Short Topics in System Administration Series

SAGE is seeking proposals from authors for booklets to be published in the series "Short Topics in System Administration." From time to time SAGE will solicit proposals for specific topics. However, SAGE is always interested in proposals covering any topic of general interest to the systems administration community.

A proposal needs to contain the following information:

In cases where SAGE is soliciting proposals for a specific topic, the first two items will be provided as part of the solicitation.

1. Working title
2. Statement of purpose: one or two paragraphs describing the purpose of the booklet
3. Draft outline
4. Name and contact information for all authors
5. Curriculum vitae for all authors
6. Representative writing sample not to exceed 500 words
7. Estimate of how long it will take to deliver the manuscript

The length of the final booklet is expected to be between 40 and 100 typeset pages. Booklets are refereed and are subject to a technical review by qualified individuals in the field. The authors will not be expected to copyedit, design, typeset, or print the booklet. An agreement regarding copyright and compensation will be executed with USENIX upon approval and acceptance of a proposal by the SAGE Series Editor.

Send proposals to the Short Topics Series Editor, William LeFebvre, at <wnl@usenix.org>. Proposals should be submitted in one of the following forms: ASCII text, html, postscript, URL reference.

About the Series

Short Topics In System Administration is a series of booklets that SAGE publishes for the system administration community. They are intended to fill a void in the current information structure, presenting topics in thorough, refereed fashion, but staying small enough and flexible enough to grow with the community. These booklets will be "living documents" that are updated as needed.

Managing Filesystem ACLs with GNU/Cfengine

Access control lists, or ACLs, are notoriously awkward to maintain. They can be a major headache for system administrators, and yet they are at the heart of POSIX filesystems and the DFS. Cfengine has begun a noninteractive approach to ACL management.

by Mark Burgess

Mark Burgess is associate professor at Oslo College. He has a PhD in theoretical physics and perhaps soon another one in computing. He is the author of cfengine. Mark claims to be interested in everything and is into getting the most out of life.

<Mark.Burgess@iu.hioslo.no>

and Demosthenes Skipitaris

Demosthenes Skipitaris has a master's degree in informatics from the University of Oslo. He has worked with UNIX systems management since 1991, and has experience with distributed filesystems like DFS. He is now partner and UNIX consultant in Nitro Media Inc.

<ds@iu.hioslo.no>

File permissions are the simplest and most basic security mechanism after passwords. Traditionally in UNIX-like operating systems, a file has one owner, one group, and a set of access bits for the owner, members of the group, and others. This model is rather old, and, for many purposes, the scheme is not flexible enough. For ordinary users, it is inconvenient to have to define a group just to open a file to a single person. ACLs are extended file permissions. They allow you to open or close a file to a named list of users without having to create a special group. They also allow you to open or close a file for a named list of groups.

Several UNIX-like operating systems have had access control lists for some time, but ACLs do not seem to have caught on – for a number of reasons in the past. The tools for setting ACLs are generally interactive, and most are very awkward to use. Because a named list of users would lead to excessive verbosity in an `ls -l` listing, one does not normally see them. Only a `+` sign next to the permission tells you that the file has a full ACL. There is therefore the danger that the information hidden in an ACL would lead to undetected blunders in opening files to the wrong users. Another problem is that ACLs are different on pretty much every vendor's filesystems, and they don't work over intersystem NFS. In spite of these reservations, ACLs are a great idea.

Here at Oslo College, users seem continually to be asking how they can open a file just for the one or two persons they wish to collaborate with. They have grown used to Novell/PC networks that embraced the technology from Apollo/NCS much earlier. The UNIX answer to users has always been: go ask the system administrator to make a special group for you. Then do the `chmod` thing. And then they would say: so what's so great about UNIX?

Addressing this lack of standardization has been the job of a POSIX Draft Committee, and some vendors have made their implementations in the image of this draft. Solaris 2.6 has a good implementation, for instance. In spite of this, even these systems have only awkward tools for manipulating ACLs – not the kind of thing you want to be around much if you have better things to do. But the incompatibility argument applies only to multiple vendor head butting. Some institutions that share data on a global basis opt for advanced solutions to network filesystems, such as AFS and DFS. DCE's DFS makes extensive use of file ACLs, and it is not operating system specific. Even so, DFS provides only interactive tools for examining and setting file permissions, and this is of little use to system administrators who would rather relegate that sort of thing to a script.

Access Control Entries

An access control list is a compound object, a bundle of information that specifies which users or groups have which permissions. The list consists of access control entries (ACEs). Each entry defines access rights to a file. Only owners or users with special authority can change the ACL. Each ACE is a set consisting of a type that tells you the kind of entry, a key that identifies the user or group for the entry, and permissions bits

associated with the key. An ACE has the form type:key:permissions. Type user defines permissions for the implied user, type group defines permissions for the implied group, and type other defines permissions for others. Some systems (like DFS) have more types (for example, type unauthenticated, which applies when the accessor does not pass authentication procedures). If the key field is empty, it means the default user/group or no user/group.

The permission bits are the standard UNIX read, write, and execute permissions. DCE/DFS has additional permission bits, control (`c`), which grants control privileges to modify the ACL, insert (`i`), which grants insert privileges, and delete (`d`), which grants delete privileges. Insert and delete refer only to directories. The ACLs also have a special mask entry. This entry indicates the maximum permissions allowed in all entries except the file owner and hence restricts the permissions specified in other entries.

Solaris ACLs are viewed with the command `getfacl` and set with `setfacl`. DFS ACLs are examined and set with the `acl_edit` command or with `dcecp -c acl <operation> <filename>`. In order to effect changes to the DFS, you have to perform a DCE login with `dce_login` to obtain authentication cookies. An ACL on Solaris may look like this:

```
# file: tt
# owner: ds
# group: iu
user::rw-
user:mark:rwx       #effective:r-x
group::r--          #effective:r--
group:sys:rwx       #effective:r-x
mask:r-x
other:r--
```

In this case, the owner of the file has `rw` permissions. User mark and group sys has `rwx` permissions, but because of the mask, the effective permissions are only `rx`. Users in the file group iu and other users has `r` permissions.

GNU/Cfengine

`Cfengine` is a freely available GNU tool for administrating nonhomogeneous distributed systems. It is class based and is designed to cope with the differences in UNIX-like operating systems in as simple a manner as possible. Although `cfengine` is not specifically a tool for administering security (which is a perceived concept), it is a tool for administering consistency. `Cfengine` makes a system converge toward a user-defined state by automating a list of key tasks such as file management and editing. One of the things `cfengine` is useful for is checking the permissions of files in a rationalized way. ACL management seemed like the kind of thing that `cfengine` should address, not least because it is desirable to port `cfengine` to modern filesystems like the DFS and to NT.

`Cfengine` is good at managing distributed environments, and many DFS users have adopted it. Interested in supporting their customers, Transarc contacted Mark and kindly offered to give us DFS so that we might add support for it in `cfengine`. We did, and the result is available from version 1.4.9, which you can pick up from GNU repositories of the from the `cfengine` home page. While we were at it, we added support for Solaris ACLs and tackled the most important problem: namely, how to make a common user interface for multiple vendor ACLs with a simple, manageable syntax.

ACL management seemed like the kind of thing that `cfengine` should address, not least because it is desirable to port `cfengine` to modern filesystems like the DFS and to NT.

The new feature in `cfengine` *is the* `acl` *directive. It is a deceptively simple looking animal, but it hides a wealth of complexity.*

Using cfengine To Manage System ACLs More Easily.

A `cfengine` bare-bones file-checking program looks like this:

```
# Free format cfengine program
control: ActionSequence = ( files )
files: classes::
/directory/file mode=644
   owner=mark,ds
   group=users,adm
   acl=zap
   action=fixall
# ... more below
```

This program checks the permissions and ownership of the named file on all machines that satisfy `classes`. The regular file mode, owner and group are specified straightforwardly as an octal number or as an `rwx` string. In `cfengine`, you can provide a list of groups or owners that you deem acceptable for a file, using a comma-separated list. If a file does not belong to any of these, then the first object in the list is taken to be the intended owner of the file. `fixall` means correct both file and directory ownership and permissions if they do not conform to this specification. The new feature in `cfengine` is the `acl` directive. It is a deceptively simple looking animal, but it hides a wealth of complexity. The name `zap`, of course, is not an access control list. Rather, `cfengine` uses a system of aliases to refer to ACLs so that the clutter of the complex ACL definitions does not impair the clarity of a file command and so that they can be reused over several file commands. An ACL alias is defined in a separate part of the program that looks like this:

```
# ...contd
acl:
{ zap
  method:append
  fstype:solaris
  user:rms:rwx
  user:len:r
}
```

Because ACLs are lists, the alias objects must also know whether the items are to be appended to an existing list or whether they are to replace an existing list. Also, because the permission bits, general options, and programming interfaces are all different for each type of filesystem, we have to tell `cfengine` what the filesystem type is.

It is possible to associate several ACL aliases with a file by adding and `acl` alias for each of these and a corresponding `acl=` option. When `cfengine` checks a file with ACLs, it reads the existing ACL and compares it to the new one. Files modification is attempted only if the files do not conform to the specification in the `cfengine` program. Let's look at a complete example:

```
files:
$(HOME)/myfile acl=acl_alias1
      action=fixall
acl:
{ acl_alias1
  method:append
  fstype:solaris
  user:len:rwx
}
```

Suppose that, before running this program, our test-file had permissions:

```
user:*:rwx
user:mark:rwx      #effective:r-x
group:*:r-x        #effective:r-x
mask:r-x
other:r-x
default_user:rw-
default_group:r--
default_mask:-w-
default_other:rwx
```

After the cfengine run, the ACL would become:

```
user:*:rwx
user:mark:rwx      #effective:r-x
user:len:rwx       #effective:r-x
group:*:r-x        #effective:r-x
mask:r-x
other:r-x
default_user:rw-
default_group:r--
default_mask:-w-
default_other:rwx
```

In cfengine, the permissions field is a comma-separated list of permissions to add, delete, or absolute assign. Addition is specified with a plus sign (+), removals with a minus sign (-), and absolute assignments, with a equals sign (=). Adding is the default action; if no sign is given, adding is assumed.

Suppose we wanted to remove the w bit for user jacobs, or make sure that it was never there. The minus sign means remove the listed permission bits.

```
{ acl_alias1
  method:append
  fstype:solaris
  user:jacobs:-w
}
```

Note that the method used here was append. That means that, whatever other access permissions we might have granted on this file, the user jacobs (a known cracker) will have no write permissions on the file. We append a delete of the write permission. Had we used the method overwrite above, we would have eliminated all other access permissions for every user and added only the above. If we really wanted to burn jacobs, we could remove all rights to the file like this:

```
user:jacobs:noaccess
```

The key word noaccess removes all bits and is equal to an absolute assignment of no permissions bits:

```
user:jacobs:=
```

Note that this is not necessarily the same as doing a -rwx, because some filesystems, like DFS, have more bits than this. Then, if we want to forgive and forget, the entry for user jacobs may be removed with:

```
user:jacobs:default
```

In Solaris, files inherit default ACLs from the directory they lie in; these are modified by the umask setting to generate their own default mask.

DFS ACLs look a little different. Suppose we have a file with the following DFS ACL:

There is nothing to stop you from creating a `cfengine` *program that is common to all the cell servers.*

```
mask_obj:r-x--
user_obj:rwxcid
user:cell_admin:r-c-        #effective:r---
group_obj:r-x-d             #effective:r-x--
other_obj:r-x--
```

Now we want to add wx permissions for user cell_admin, add new entries with rx permissions for group acct-admin and user `root`, and remove the `x` bit for the file owner. This is done with the following ACL alias:

```
{ acl_alias2
  method:append
  fstype:dfs
  user:/.../iu.hioslo.no/cell_admin:wx
  group:/.../iu.hioslo.no/acct-admin:rx
  user:/.../iu.hioslo.no/root:rx
  user:*:-x
}
```

The local cell name `/.../iu.hioslo.no` is required here. `Cfengine` cannot at this time change ACLs in other cells remotely unless it is started from a shell that already has cross-cell authenticated rights. However, there is nothing to stop you from creating a `cfengine` program that is common to all the cell servers (the same file parsed by each machine). Each server would then read and execute the same file, and each would look after its own cell. The job is then automatically distributed without the need for cross-cell authentication. This is the beauty of `cfengine`. After running `cfengine` with the above program snippet, the ACL becomes:

```
mask_obj:r-x--
user_obj:rw-cid
user:cell_admin:rwxc-       #effective:r-x--
user:root:r-x--             #effective:r-x--
group_obj:r-x-d             #effective:r-x--
group:acct-admin:r-x--
other_obj:r-x--
```

For the sake of simplicity, we have used only standard UNIX bits `rwx` here, but more complicated examples may be found in DFS, for example,

```
user:ds:+rwx,-cid
```

which sets the read, write, execute flags but removes the control, insert, and delete flags. In this case, this is equal to:

```
user:ds:=rwx
```

In the DFS, files inherit the initial object ACL of their parent directory while new directories inherit the initial container object.

The objects referred to in DFS as `user_obj`, `group_obj`, and so forth refer to the owner of a file (i.e., they are equivalent to the same commands acting on the user who owns the file concerned). To make the `cfengine` user-interface less cryptic and more in tune with the POSIX form, we have dropped the `_obj` suffices. A user field of `*` is a simple abbreviation for the owner of the file and an empty key field.

The Future

A problem with any system of lists is that one can generate a sequence that does one thing and then undoes it and then redoes something else, all in the same contradictory list. To avoid this kind of accidental interaction, `cfengine` insists that each user has only one ACE (i.e., that all the permissions for a given user be in one entry).

Several people have said it would be desirable to port `cfengine` to Windows NT. Although lack of time is the main reason this has not yet happened, we agree that this is just what NT is missing. The ACL model we have chosen would work equally well there and would be even more important, because NT does not have a simple `chmod`-like default permission. This is clearly an important project for the future.

As of now, we have implemented the ACL code for Solaris filesystems because the latter is supposed to be essentially the POSIX draft) and for DFS and have left stubs for people to add their own ports. Other system types will come as we get time. Perhaps the enthusiastic `cfengine` community, having been provided with the infrastructure, will come up with ACL code for the most-used operating systems. It would certainly accelerate the pace.

Access control lists are a valuable addition to the UNIX repertoire. They are useful for aiding collaboration among users and implementing tighter security. It would be nice if vendors woke up and standardized ACLs so that we could all use them over the NFS. It would also be nice if Linux and the BSDs had ACLs. For now, `cfengine` will allow system administrators to reap their benefits on the systems that have them.

It would be nice if vendors woke up and standardized ACLs so that we could all use them over the NFS. It would also be nice if Linux and the BSDs had ACLs.

References

`Cfengine` homepage <http://www.iu.hioslo.no/cfengine>.

M. Burgess, "A Site Configuration Engine," *Computing Systems* 8 (1995), p. 309.

M. Burgess and R. Ralston, "Distributed Resource Administration Using cfengine," *Software Practice and Experience* 27(9), (1997), p. 1083.

Sun Microsystems, *SunOS Reference Manual* (User commands).

DFS documentation. Transarc Corporation.
<http://www.transarc.com/afs/transarc.com/public/www/Public/Documentation>

interview with Dan Hildebrand

Dan Hildebrand <danh@qnx.com> is a senior architect at QNX Software Systems Ltd. (QSSL), where he plays a role in product steering, implementation, and the various miscellany for which medium-sized organizations are so well known. Rob Kolstad interviewed Dan electronically during March 1998.

Rob: You're at QNX. Tell us a little about your company and its recent history.

Dan: QSSL has been developing and marketing a microkernel, realtime OS for many years now, and we've managed to sell over a million copies. Most people use QNX several times a day now without being aware of it. Although industrial automation and the medical instrumentation industry have been good for us, recently, we've generated a lot of consumer appliance design wins in the form of Internet set top boxes and similar devices. We expect to see three million set top boxes running QNX over the next 12 to 18 months.

Rob: So, you're based on FreeBSD or one of the other BSD-based operating systems?

Dan: Not at all. QNX is a from-scratch implementation of a microkernel, realtime OS that implements the POSIX API (for which we passed the compliance certification test). Although UNIX and POSIX source code ports to QNX as easily as to any other UNIX variant, architecturally, QNX provides a true microkernel architecture. That architecture enables network-transparent, fault-tolerant distributed computing and various other attributes that you don't typically see on monolithic kernel OSs.

Rob: As a realtime OS, it probably has some special technology in the kernel, then, for realtime and other embedded customers.

Dan: A long-standing capability in QNX has been realtime and embedded services. The implementation of any realtime OS requires a continuous attention to those goals. There are all kinds of ways that nonrealtime OSs don't honor priority inheritance or account for priority inversion or simply don't remain modular enough to scale down for memory constrained embedded systems. So, unless you're vigilant throughout the entire development process, you can easily end up with a nonrealtime, nonembeddable OS.

Rob: So what's implemented in the microkernel?

Dan: Three fundamental classes of service in 16 kernel calls: message passing, process scheduling, and first-level interrupt handling. Everything else is implemented by optional processes that can be started and stopped at runtime and are scheduled for execution by the microkernel. As a result, the system is completely open and end-user extensible. This approach also means that there aren't any services in QNX that aren't directly implementable by the developer, using only the published OS API.

Rob: There are so many stories about microkernels, probably mostly from the Mach project. Are you seeing performance problems?

Dan: Nope. QNX first came out in the early eighties and we've been tuning and tweaking our architecture for a long time now. Hardly a month goes by that we don't find another tweak to speed things up or make things just a bit smaller. How we implement context switches, how the TLB tables are arranged, etc. are under almost constant scrutiny. Making context switches and message passing quick on QNX makes it very natural to split an application into a team of cooperating processes and let QNX handle the IPC between them. Because they are separate processes, the robustness issues of memory protection become applicable, and the application also distributes across multiple processors on a QNX LAN very naturally because all IPC under QNX is network transparent.

Rob: How much of QNX lives out in user-space?

Dan: Virtually everything. Device drivers, interrupt handlers, network services, etc. Having achieved very inexpensive IPC between user-space processes, we don't have to pull services into the kernel to address performance issues. The only services that end up in the kernel become those with a legitimate need (performance having been addressed otherwise).

Rob: Does the microkernel use up appreciably more code or data space than comparable monolithic kernels?

Dan: I think this is a "quality of implementation" comparison. Some microkernels will be smaller; some monolithic kernels will be smaller, depending upon the quality of implementation. In the case of QNX, you'll be very hard-pressed to find a monolithic kernel that delivers comparable functionality in a similar memory footprint. Besides, I wouldn't say the point of a microkernel OS is necessarily small size. It's more about a different architectural partitioning of the OS in order to address other functionality requirements (even though we did our demodisk (downloadable from <www.qnx.com>) as an example of that functionality vs. memory size issue. Incidentally, no one in the Linux (or BSD) community has equalled that demodisk, although they continue to try.

Rob: I've always wondered – did you have any particular problems implementing networking protocol stacks?

Dan: Nope. Because we created an efficient user-space interrupt handler architecture, our network drivers and protocol suites run quite naturally as user-space processes that attach to the necessary hardware to move packets around. We achieve wire rate quite naturally, but can start and stop networking services as easily as any user-space process.

Rob: What's your favorite feature of microkernels?

Dan: It's hard to narrow it to one (ease of development is certainly significant to an OS developer), but if I have to name only one, it would be robustness. Because all OS services and applications are implemented as separate, MMU-protected processes, stray pointer errors in even OS-level processes, or device drivers can't bring the system down. A "software watchdog timer" can readily restart failed portions of the system. Also, because OS modules run in a binary-identical fashion from system to system, without needing to be relinked for reuse in a new runtime environment, the quality experience you may have with a given process remains applicable even in new system configurations. For focusing the efforts of multi-hundred-person development teams into a single QA'd development environment for embedded systems, this attribute is invaluable.

Rob: Do you have any particular pet peeve?

Dan: Oh yeah. As with any class of software, there are good implementations and bad implementations. Unfortunately, because some microkernel implementations don't perform as well as monolithic kernels, many people generalize from this and assume that all microkernels are slower than all monolithic kernels. Microsoft's marketing folks calling NT a microkernel OS certainly doesn't help this. QNX serves as a good example of what a microkernel architecture, properly implemented, can do.

Rob: Do you think we'll be seeing more microkernels in the future?

Dan: As a platform for OS experimentation, a microkernel can't be beat. It's trivial to run experimental filesystems beside stable filesystems and do interesting OS development. This sort of development can be painful with a monolithic-kernel OS. In fact, some microkernel OSs are used as microkernels during development, but not deployed

Dan Hildebrand

"As a platform for OS experimentation, a microkernel can't be beat. It's trivial to run experimental filesystems beside stable filesystems and do interesting OS development. This sort of development can be painful with a monolithic-kernel OS."

that way for runtime. I would suggest that that misses the point. Many of the advantages of microkernel OSs are valuable at runtime – it's up to the OS designer to make sure the implementation performs well so that the architectural benefits of the microkernel remain useful at runtime. However, I would suspect that high-performance, low-overhead microkernels are harder to implement well, and that's why there are fewer good microkernels out there. To answer the question, I expect there will be more microkernels OSs appearing over time, especially as products that benefit from the technology become more common.

Rob: Are there any other technologies you see on the horizon that bear watching?

Dan: Bandwidth and connectivity. Wireless or connected. Except for the computational fringe and the ultra-low-cost embedded marketplace, most people have an approximately adequate amount of CPU horsepower for their applications. However, any processor that communicates with another could almost invariably demonstrate an improvement in functionality if it had more bandwidth. Because Internet communications technologies are appearing in virtually every embedded system these days, the appetite this creates for more bandwidth will make it the most rapidly evolving aspect of our computing environment for the next decade. Of course, higher and higher bandwidth increases the pressure on OS designers to deliver that bandwidth efficiently to those communicating applications. That remains the OS architect's challenge.

Rob: Thanks for your time!

Dan: You're welcome.

use the source, Luke! again

So you call yourself a UNIX hacker: you know what `bread()` is, and the various `splxx()` routines don't faze you. But are you *really* a UNIX hacker? Let's have a look at a brief history of UNIX and the community of UNIX users and hackers that grew up around it and some recent developments for real UNIX hackers.

By Warren Toomey

Warren Toomey is a lecturer in computer science at the Australian Defence Force Academy, where he just finished his Ph.D. in network congestion. He teaches operating systems, data networks, and system administration courses. He has been playing around on UNIX since 4.2BSD.

<wkt@henry.cs.adfa.oz.au>

Editor's note: This article originally appeared in a slightly different form in the AUUG Newsletter.

UNIX took the academic world by storm in 1974 with the publication of Ken Thompson's paper about its design, which was published in *Communications of the ACM*. Although it didn't contain many radically new ideas, UNIX had an elegance, simplicity, and flexibility that other contemporary operating systems did not have. Soon lots of people were asking Bell Laboratories if they could get copies of this wondrous new system.

This was the cause of some concern within AT&T, because of the restrictions of an antitrust decree brought against them in the 1950s. This decree effectively stopped AT&T from selling or supporting software: they could only engage in telco business. Their solution to meet the UNIX demand was to charge a nominal "license" fee to obtain UNIX and to distribute tapes or disks "as is." You'd receive your disk in the mail with just a short note: "Here's your rk05. Love, Dennis."

AT&T's stance on UNIX was often seen as an OHP slide at early conferences:

No advertising
No support
No bug fixes
Payment in advance

"This slide was always greeted with wild applause and laughter," says Andy Tanenbaum. This lack of support was tolerated for several reasons: Ken and Dennis did unofficially fix things if you sent them bug reports, and you also had the *full source code* to UNIX.

At the time, having full source code access for a useful operating system was unheard of. Source code allowed UNIX users to study how the code worked (John Lions's commentary on the sixth edition), fix bugs, write code for new devices, and add extra functionality (the Berkeley Software Releases, AUSAM from UNSW). The access to full source code, combined with AT&T's "no support" policy, engendered the strong UNIX community spirit that thrived in the late 1970s and early 1980s, and brought many UNIX users groups into existence. When in doubt as to how a program (or the kernel) worked, you could always "use the source, Luke!"

During this period, UNIX became wildly popular at universities and in many other places. In 1982, a review of the antitrust decree caused the breakup of AT&T into the various "Baby Bell" companies. This gave AT&T the freedom to start selling software. Source code licenses for UNIX became very expensive, as AT&T realized that UNIX was indeed a money spinner for them. Thus the era of UNIX source code hackers ended, except for some notable activities like the 4BSD work carried out at the University of California, Berkeley.

Those organizations lucky enough to have bought a "cheap" UNIX source license before 1982 were able to obtain the 4BSD releases from UCB and continue to hack UNIX. Everybody else had to be satisfied with a binary-only license and wait for vendors to fix bugs and add extra functionality. John Lions's commentary on how the UNIX kernel worked was no longer available for study; it was restricted to one copy per source code license, and was not to be used for educational purposes.

UNIX hackers of the late 1990s surely have an abundance of source code to hack on: Linux, Minix, OpenBSD, etc. But are they really UNIX hackers, or just UNIX clone hackers? Wouldn't it be nice if we could hack on real UNIX, for old time's sake?

What were UNIX hackers going to do with no UNIX source code to hack anymore? The solution was to create UNIX clones that didn't require source code licenses. One of the first was Minix, created by Andy Tanenbaum and aimed squarely at teaching operating systems. Early versions of Minix were compatible with the seventh edition UNIX; the most recent version is POSIX compliant and can run on an AT with 2 MB of memory and 30 MB of disk space.

Many Minix users tried to convince Andy to add features such as virtual memory and networking, but Andy wanted to keep the system small for teaching purposes. Eventually, a user named Linus Torvalds got annoyed enough that he used Minix to create another UNIX clone with these extra features. And so Linux was born.

While Linux was taking off like a plague of rabbits, the BSD hackers were working on removing the last vestiges of UNIX source code from their system. They thought they had done so, and BSDI released BSD/386, a version of 4.3BSD that ran on Intel platforms. AT&T, however, wasn't so sure about the complete removal of UNIX source code and took them to court about it.

AT&T is not a good company to be sued by: it has a small army of lawyers. Eventually, the conflict was settled out of court with a few compromises, and we now have several freely available BSDs: FreeBSD, NetBSD, and OpenBSD. Of course, they all come with source code.

UNIX hackers of the late 1990s surely have an abundance of source code to hack on: Linux, Minix, OpenBSD, etc. But are they really UNIX hackers, or just UNIX clone hackers? Wouldn't it be nice if we could hack on real UNIX, for old time's sake?

UNIX turned 25 in 1993, which makes its early versions nearly antiques. Many of the old UNIX hackers (hackers of old UNIX, that is) thought the time had come to get the old, completely antiquated UNIX systems back out for sentimental reasons. After all, ITS, CTSS, and TOPS-20 had been rescued and made publicly available, why not UNIX?

At the time, UNIX was undergoing a crisis of ownership. Did AT&T own UNIX this week, or was it Novell, Hewlett-Packard, or SCO? UNIX is a trademark of someone, but I'm not sure who. After the dust had settled, SCO had the rights to the source code, and X/Open had dibs on the name "UNIX," which is probably still an adjective.

During the ownership crisis, Peter Salus, Dennis Ritchie, and John Lions had begun to lobby Novell: they wanted John's commentary on UNIX to be made publicly available in printed form. It wasn't until the UNIX source code rights had been sold to SCO that this finally was approved. It helped to have some old UNIX hackers, Mike Tilson and Doug Michels, inside SCO to fight the battle. You can now buy John Lions's commentary on 6th Edition UNIX (with source code) from Peer to Peer Communications, ISBN 1-57398-013-7. As Ken Thompson says: "After 20 years, this is still the best exposition of a 'real' operating system."

One of the restrictions on the commentary's publication is that the UNIX source contained within cannot be entered into a computer. OK, so you can read the book, but what use is source code unless you can hack at it?!

At the time that SCO bought UNIX, I began to lobby SCO to make the old source available again, unaware of the efforts to release the Lions's commentary. SCO's initial

response was "this will dilute the trade secrets we have in UNIX, and it wouldn't be economically viable." My efforts drew a blank.

To help bring greater lobbying power to bear on SCO, the PDP UNIX Preservation Society (PUPS) was formed. Its aims are to fight for the release of the old UNIX source, to preserve information and source from these old systems, and to help those people who still own PDP-11s to get UNIX up and running on them. After realizing that SCO was never going to make the old UNIX source code freely available, we explored the avenue of cheap, personal-use source licenses. The society set up a Web petition on the topic and gathered nearly 400 electronic signatures.

Inside SCO, we were very fortunate to contact Dion Johnson, who took up our cause and fought tooth and nail with the naysayers and the legal eagles at SCO. The combined efforts of the PUPS petition and Dion's hard work inside SCO has finally borne fruit.

On March 10, 1998, SCO made cheap, personal-use UNIX source code licenses available for the following versions of UNIX: first through seventh edition UNIX, 32V, and derived systems that also run on PDP-11s, such as 2.11BSD. The cost of the license is US$100, and the main restriction is that you cannot distribute the source code to people without licenses. Finally, we can be real UNIX hackers and "use the source, Luke!" again.

Finally, we can be real UNIX hackers and "use the source, Luke!" again.

Acknowledgments and References

I'd like to thank Dion Johnson, Steven Schultz, the members of the PDP UNIX Preservation Society, and the people who signed the PUPS petition for their help in making cheap UNIX source licenses available again. Dion, in particular, deserves a medal for his efforts on our behalf.

You can find more about the PDP UNIX Preservation Society at <http://minnie.cs.adfa.oz.au/PUPS/> and details on how to obtain your own personal UNIX source license at <http://minnie.cs.adfa.oz.au/PUPS/getlicense.html>.

SCO won't be distributing UNIX source code as part of the license. PUPS members have volunteered to write CDs and tapes to distribute old versions of UNIX to license holders. We currently have fifth, sixth, and seventh editions, 32V, 1BSD, all 2BSDs, Mini UNIX, and Xinu. We are looking for complete versions of PWB UNIX and AUSAM. We desperately want anything before fifth edition and hope these early systems haven't gone to the bit bucket. Please contact us if you have anything from this era worth preserving.

If you are licensed and want a copy of the PUPS Archive, see the PUPS Web page above for more information. We expect to be deluged with requests for copies, so if you can volunteer to write CDs or tapes for us, please let us know.

You don't need own a PDP-11 to run these old systems. The PUPS Archive has a number of excellent PDP-11 emulators. If you have bought a copy of the Lions's commentary (and you should), now you can run real sixth edition UNIX on an emulator. And if you want, you can hack the code!

PC hardware for source code UNIX

The Double-Edged Sword

By Bob Gray

Bob Gray is co-founder of Boulder Labs, a digital video company. Designing architectures for performance has been his focus since he built an image processing system on UNIX in the late 1970s. He has a PhD in computer science from the University of Colorado.

<bob@boulderlabs.com>

Introduction

On the one hand, the enormous marketing pressures of hundreds of PC hardware vendors forces rock bottom prices. On the other, only a small fraction of the thousands of combinations are well balanced. Although there are plenty of high-performing hardware devices, many have marginal value due to design flaws, cost constraints, poor reliability, or inadequate drivers. I explore many of the issues surrounding choosing PC hardware and make some specific recommendations focused on running source code UNIX. This article deals with the hardware components at the "executive" level; see the Web references for details, depth, and additional recommendations.

I discuss components in terms of three levels of target systems: low, medium, and high. I'll define a "system" as a CPU, motherboard, disk, memory, CD, floppy, Ethernet, video/graphics card, keyboard, mouse, power supply, and case. Add a monitor and you have a complete workstation.

The low system aims for good performance at the lowest possible price. I forgo some upgradability and expansion capability in this system. I get the cost advantage of one- or two-year-old technology. The system, which will be a respectable performer, will cost around $800 to $1,200 (without a monitor).

The medium system is a very substantial general-purpose workstation. By increasing the budget to $1,500 to $2,000, I can get a motherboard that will take current generation CPUs, faster memory, and a better disk subsystem. The technology here is about three to nine months old. I will be able to upgrade and expand these systems with bigger/faster disks, lots more fast memory, and faster CPUs.

In the high system I will pay a premium for everything, spending $2,200 to $3,500 to have the latest, widely available hardware. Some of these components are just being released. I may find myself helping debug new drivers for this equipment, but be assured, this system will scream!

For your convenience, an HTML version of this article, with active Web references, is at <www.boulderlabs.com>.

Organization

This article dissects a PC system component by component. (See April 1998 *;login:* for the motivation to run UNIX on your PC.) I freely give my advice and opinions on what works and what is valuable. I intersperse Web references that will allow you to get more details. The prices mentioned have been found on the Web in one or more places – they are not always the lowest. At the end of the article, I mention several integrators; give these guys a list of what you want, and they will quote a system price and maybe even load software for you. You could also purchase the individual components and assemble the system yourself. Look at <www.computeresp.com>, <www.pricescan.com>, and the other references in this article for price shopping. Enjoy.

Cases and Power Supplies

You've got to have something to hold, power, and cool all the pieces. Although most any case and power supply will sort of work, there are reasons to be selective. I prefer a case that has a simple single removable side panel or two removable side panels for convenient access. Over the years, I have found the wrap-around style cases a major

nuisance. You'll want to think about how big or small of a case to buy. How many I/O peripherals (e.g., tapes, CD-ROMS, DVDs, and disks) will you be connecting? I strongly recommend the ATX-style case, which requires a different kind of power supply. It gives you onboard serial connectors and PS/2-style mouse/keyboard connectors. Even for the low system, I will use an ATX-style case. Most motherboard manufacturers are going over to the ATX form factor. So if you get an ATX case and power supply, you'll have more options when you want to upgrade again. I like the Personal Mid-Tower ATX Enclosure for $69. See <www.pcpowercooling.com> and <www.american-media.com/gigastar.html>. See <www.tdl.com/~netex> for a discussion of other options for a chassis. Also, California PC Products, <www.calpc.com>, makes quality cases.

Cheap power supplies will cost about $35. The better supplies will have higher quality voltage regulation and a longer lasting, ball-bearing fan with a two-year or longer warranty. I like the quiet operation of the Silencer 235 ATX for $79. If you need lots more power or cooling, look at the Turbo-Cool 450 and 600 which have a five-year warranty and MTBF of 70,000 to 100,000 hours respectively (see <www.pcpowercooling.com>).

Central Processing Unit

At the heart of the workstation is the Central Processing Unit (CPU), which for me must be x86 compatible. Intel makes the Pentium, Pentium-Pro, and Pentium-II. Pentium chips' main competitors, AMD's K6 line and Cyrix's 6x86MX line, generally cost less. They are fully compatible and should be trouble-free. The issues are described in more detail on Tom's hardware Web page: <www.tomshardware.com>.

The classic Pentium is pretty much at the end of its lifetime. Pentiums with MMX are also thinning out, but still available for about $100 for a 166MHz chip with a fan and a heat sink. MMX instructions themselves can be useful if you're willing to write assembly and have specific applications in mind. Most contemporary games will take advantage of the MMX instructions. Even though most UNIX applications do not use the MMX instructions, you get a 16K instruction and 16K data L1 cache, which makes the MMX option worthwhile for general-purpose computing. A Pentium/MMX 233MHz costs about $200, or you could get the middle Pentium/MMX 200MHz. The connection to the motherboard is called "Socket-7."

The Pentium Pro 200 is also near end of its lifetime, but still available. It will give you three instruction pipelines, 8K L1 instruction cache, and 8K L1 data cache, along with L2 cache in sizes of 256K, 512K, or 1M. With its L2 cache that runs at CPU speed, some applications may run 50% faster than on a Pentium with the same clock speed. I have heard a few stories of a Pentium Pro 200 running compute-intensive applications faster than a Pentium II 300MHz. It's likely because the Pentium Pro cache runs at processor speed and the Pentium II's cache runs at one-half the processor speed. The 256K L2 cache versions go for about $325. The 512K version costs $750; but you may find deals on the spot market.

The Pentium II is Intel's new CPU line. It requires a motherboard with a "Slot 1" connection. The big win for it is the ability to use SDRAM (see the "Memory" section later in this article). The Pentium II has 16K L1 instruction cache and 16K data cache. The Pentium II can cache only 512MB of RAM. See <www.intel.com/pentiumII/specs/fact.htm>. The following Pentium II CPUs have been available for several months:

233/66/512 for $300
266/66/512 for $400
300/66/512 for $550
333/66/512 for $625

Even though most UNIX applications do not use the MMX instructions, you get a 16K instruction and 16K data L1 cache, which makes the MMX option worthwhile for general-purpose computing.

The first issue for choosing a motherboard is deciding what CPU you will be using.

The first number is the processor speed in megahertz, the second is the system bus speed in megahertz, and the third is the size of the L2 cache in kilobytes.

Just released in April and important for high-end systems are the Pentium II 350/100/512 and the 400/100/512. These new CPUs can cache 4GB of RAM. Notice that the system bus speed has increased to 100MHz. You'll need a newer motherboard to be able to take advantage of the higher system bus speed.

Expected to be released in June or July is the Pentium II called "Deschutes." It will run at 100MHz bus speed and come with a so-called "CSRAM" second level cache that will run at CPU clock speed rather than one-half CPU clock in the Pentium II. The Deschutes's cache will be able to access up to 4GB RAM. This CPU will need a new connection, called "Slot 2." See <www.thechipmerchant.com> for processors and memory.

Motherboards and Chipsets

The first issue for choosing a motherboard is deciding what CPU you will be using. You will want to look at the bus speed supported and on board peripherals such as SCSI and Ethernet. I recommend only the ATX form factor; it puts the serial and parallel connectors on board instead of on extra, problematic cables. Some folks will want to run motherboards with multiple processors. Although this article doesn't address it, Symmetric MultiProcessing (SMP) works well with the right hardware and software. Intel provides the following motherboard chipsets:

430TX Pentium chipset
30HX Pentium chipset
440FX Pentium Pro/Pentium II chipset
450GX Pentium Pro chipset
440LX Pentium II chipset
440EX Pentium II chipset (lightweight version of 440LX)
440BX w/ PIIX4 Pentium II chipset
450NX Deschutes Slot 2 chipset (due out this summer)

The old 430TX and the 430HX are the Socket-7 workhorses. Although they limit cacheable memory to 64MB, they are a great deal for low systems.

The 440LX Pentium II chipset comes with AGP (see "Video Graphics" later in this article) and SDRAM support to increase the performance of Pentium II systems. It supports a bus speed of 66MHz. This is the chipset you want if you need a high-performing system today at an economical price. It should serve you well for quite some time.

The 440BX Pentium II chipset will finally give you the 100MHz system bus speed. Expect to pay a premium for this new, high system bus chipset, but it should be worth it for high-end systems.

The 450NX Deschutes Slot 2 chipset will finally replace the 450GX Pentium Pro chipset for server platforms. This chipset will be the first for Slot 2 and hence run only with the Slot 2 Deschutes CPU. It will support up to quad CPU systems. This chipset will run at 100MHz bus clock and is designed for server systems.

Other chipsets include Acer Labs Inc. (ALi), Silicon Integrated Systems, and VIA. See <www.tomshardware.com> for more information.

There are many motherboard manufacturers incorporating the previously mentioned chips. You can check the references for details or just go with the recommendations I give below. For a given motherboard, you'll want to know how many CPUs it handles, the kind and amount of memory (SDRAM, EDO, FPM, etc.), the system bus speed, the form factor (ATX or AT), and the number of PCI and ISA slots. Also look for what

peripherals are supported onboard. Finally, you may want to know about the BIOS supplied and whether it is flashable. See the BIOS section on <www.tomshardware.com>. Also see <www.anandtech.com> for numerous motherboard reviews.

The solid, inexpensive ($150) ASUS P/I-XP55T2P4 motherboard runs the 430HX chipset and supports Pentium, Cyrix, and AMD-K5 CPUs with Socket-7. It is a great candidate for a low system.

The FIC PA-2012 ($100) has a 1MB cache and AGP support for Socket-7. This board will run the Pentium, AMD-K6, and Cyrix 6x86MX. Further, it supports SDRAM. Put in an AMD-K6 233 ($120) processor, load it up with SDRAM, and you have a very hot, inexpensive system (see <www.fic.com.tw>).

For the Pentium Pro, consider the ASUS P/I-P65UP5 with C-P6ND CPU card (two 200MHz P6s, 256K). One of the reviewers runs this as his main system; it has been fast and reliable. The motherboard has 8 SIMM slots onboard, and you can get up to 512MB. It also has five PCI slots and three ISA slots. You can have a total of only seven cards, though, because one of the PCI and one of the ISA slots share the same space. The CPUs are on a daughter card, and you can get a dual Pentium daughter card for it (C-P55T2D) or the dual Pentium Pro card (C-P6ND). See <www.asus.com.tw/Products/Motherboard/Pentiumpro/P65up5-pknd/p65up5-pknd-spec.html>.

The ASUS P2L97-S for $250 is an excellent choice for a Pentium II system using Slot-1. It has an integrated SCSI using the same AIC-7880 chip as on the Adaptec 2940UW. This board takes SDRAM and handles Parity/ECC. The P2L97-DS will handle dual Pentium IIs.

The Intel DK440XL motherboard ($600) is a big win if you plan to run a Pentium II. Onboard, it includes the Adaptec AIC-7895 SCSI chip and an Intel 82557, 10/100baseT chip – two fewer PCI boards to buy. You'll need to run very current software to take advantage of the very new 7895 chip. This motherboard can handle dual Pentium IIs. This board takes SDRAM and handles Parity/ECC; see <developer.intel.com/design/motherbd/dk/index.htm>.

The Intel PR440FX, which handles dual Pentium Pros, has both 10/100 Ethernet and ultrawide SCSI onboard, but only one ISA slot.

Memory

I am a strong advocate of reliable memory. I frown on those who choose to run without parity or ECC. When things go wrong, I want as much help as possible in pinpointing the problem. Historically, PCs have not had good memory subsystems, but that is changing. You can get very fast memory today, along with full ECC protection. (Error correcting codes allow the correction of one bit error per memory word and detection of up to two bit errors per memory word. If you ran parity detection, two bit errors in the same memory word would be a wash, and the hardware would not report a problem.) You need the right kind of memory SIMMs (DIMMs, etc.) and the right motherboard to support parity or ECC. Generally, you can enable these options in BIOS settings.

When buying RAM, look specifically for the features "Parity" and "ECC" in the components, and stick with the more reputable organizations that will ensure that their memory will work in your system. (Some of this new SDRAM memory is borderline.) The new SDRAM is considerably faster than the previously available memory. One of the reviewers ran a trivial memory test ("dd if=/dev/zero of=/dev/null bs=1024k count=1024") to read and write a gigabyte. He saw about a 50% improvement in speed

I am a strong advocate of reliable memory. I frown on those who choose to run without parity or ECC. When things go wrong, I want as much help as possible in pinpointing the problem.

Manufacturers are rapidly introducing bigger disks and dropping their smaller products.

by using SDRAM over FPM: 100MB/s for DRAM PP FPM and 150MB/s for SDRAM on a PII. A 64MB SDRAM 8x72 parity/ECC module costs about $225. You can look at John McCalpin's benchmarks for more memory tests at <www.cs.virginia.edu/stream>. There are also memory tests in Larry McVoy's "lmbench" tests available in the benchmark ports collections, (<www.freebsd.org/ports/benchmarks.html>). A reputable supplier of memory is <www.thechipmerchant.com>.

I/O Bus

I am a fan of SCSI-based I/O. Although IDE disks are cheaper and you don't have to buy a separate controller board, I recommend spending the extra money if you want more reliability or more performance. SCSI gives you a lot more flexibility for connecting devices to your computer, and it gives you the efficiency of "Tagged Command Queuing" (multiple outstanding requests that the drive's firmware can reorder for efficiency). You can have multiple drives seeking while another is transferring. It is easy and efficient to share several disks, a CD-ROM, and tapes on an SCSI bus. You get only two devices per IDE bus, and a transfer on one locks out the other. Although pricey, my favorite SCSI bus controller board is the Adaptec 2940UW for about $220. You can hook up both narrow and wide devices at the same time and run at ultraspeed (see below). Make sure you properly terminate the SCSI bus at the end by using the disk termination jumper or an SCSI active terminator. Some of the new motherboards incorporate SCSI onboard by using the same Adaptec 7880 chip as is in the 2940UW. This saves the cost of a board and a PCI slot. A somewhat less expensive controller card is the Buslogic (now Mylex).

Disk

The consensus is that SCSI disks are more reliable than IDE drives. Many SCSI drives come with five-year warranties; IDE drives typically have warranties of three years or less. Don't buy a disk without at least a three-year (preferably five) warranty. Typical IDE drives today spin at 5,400 RPM; SCSI disks provide 7,200 RPM performance.

There are two significant bus widths: the "narrow" 1-byte, 50-pin kind and the "wide" 2-byte, 68-pin kind. You need the right kind of cables to connect these devices. The Adaptec 2940UW has both kinds of connectors so you can easily mix narrow and wide devices.

The two predominant bus speeds in use today are "fast" (10MHz) and "ultra" (20MHz). (Just recently available is the Ultra-2 LVD at 40MHz: 80MB/s.) So a fast-wide SCSI bus can move up to 20MB/s and ultrawide can move up to 40MB/s. (With three high-performance disks going at once, I have measured more than 17MB/s aggregate on fast-wide, so I would expect an aggregate of into the 30s for ultrawide.)

Now, what width of SCSI drives do you buy (50pin or 68pin)? If you will have only one or two disks on your SCSI bus, then you don't need the ultrawide version. But if you are building a server with many disks or high-performance disks, you probably want the extra head room of 40MB/s SCSI ultrawide. Copying from one disk to another will require more that 20MB/s aggregate if you are using the new 10,000 RPM disks (see IBM 9ZX below). Although the traditional workstation vendors like SUN and SGI have had Fibre channel to disks or disk towers for a couple of years, I have not seen much FC activity for PCs. Either 80MB/s SCSI will catch on or 100MB/s FC will catch on.

Manufacturers are rapidly introducing bigger disks and dropping their smaller products. For example, last summer Seagate phased out its 1GB SCSI Hawk drives. Last fall

they phased out their 2GB SCSI Hawk drives. You can still buy 2GB Quantum Atlas II disks, but not for long. On the high end, you can buy up to 23GB disks. I wonder how long 4GB drives will last. These changes make for great deals on the spot market. Check with <www.pricewatch.com> for specials.

I like the Seagate Barracuda product line (<www.seagate.com>). These are solid, high-performance 7,200 RPM drives with five-year warranties.

ST34371N Barracuda 4GB Ultra $550
ST34371W Barracuda 4GB Ultra Wide $600
ST19171N Barracuda 9GB Ultra $790
ST19171W Barracuda 9GB Ultra Wide $820
ST34501N Cheetah 4GB Ultra $630
ST19101W Cheetah 9GB Ultra Wide $999

The higher end drives include the IBM Ultrastar 9ZX spinning at 10,000 RPM and the Seagate Cheetah 4LP. These have faster access times, faster transfer rates, and bigger buffers. IBM claims that its can do 17MB/sec sustained. (See <www.storage.ibm.com/hardsoft/diskdrdl/ultra/9zxdata.htm>). These days, IBM seems to be interested in providing drives to the general market. In the past, only its surplus was available. It will be good if IBM becomes a reliable supplier – Ultrastar drives are among the best on the market today.

The Quantum Atlas II is also a good drive. Its features include 7,200 RPM, 4.5GB or 9.1GB, 8.0 ms Average Seek Time, and a five-year warranty.

Modem

The primary consideration with modems is that they be compatible with your Internet service provider. If you are lucky enough to be in an area where 56Kb modems will work, then you can go with either a KFlex or X2-style modem. In early 1998, the ITU standardized on V.90 for 56Kb modems. Most contemporary modems can be updated (FLASH memory) to be V.90 compatible, but again, check what your service provider recommends.

Some of us like external modems because they can be reset without resetting the computer. You can see what is happening with all of their status lights. Others prefer internal modems because they are cheaper, don't need a power supply, have fewer wires, and don't take up a serial port.

If you intend on working with faxes under UNIX with software like Hylafax, make sure you get a class two modem. This kind, unlike class one, does most of the protocol negotiation in the modem instead of requiring your software to do it. Avoid the nameless modems – they can cause you lots of headaches. The following are "name brand" modems that should serve you well:

Zoom/FaxModem 56K external K56Flex ($116)
US Robotics External Sportster FAX Modem ($150)

Video/Graphics

The most troublesome area for running UNIX on PCs has to do with supporting the video/graphics cards. Before you buy that super hot Voodoo2 card to play DOOM or other games, check what kind of support is available for X11. On a workstation that doesn't have heavy 3-D requirements, the good old Matrox Millenium II 4MB for $210 is an excellent choice. You can run this one up to 1280x1024 by 24 bits. If you want good 1600x1200 resolution or 1800x1440, get 8MB of video memory. The game players

The most troublesome area for running UNIX on PCs has to do with supporting the video/graphics cards.

If you are working for someone, make them understand that you will be happier and more productive if you are looking at a sharper, clearer, no-flicker screen. The economics are compelling.

will want to look at the new 3-D accelerated boards. Check the PC magazines and <www.tomshardware.com> for hints. Beware, the XFree86 Project lags behind for some of these boards. Check <www.xfree86.org> to find the currently supported list. The folks at <www.suse.de/XSuSE/XSuSE_E.html> use XFree86 as a base and support additional cards. Finally, Xi Graphics, Inc. supports many of the most modern boards. For a modest price, this may be the way to go. See <www.xig.com>. Currently, AGP graphics doesn't buy you much performance over PCI graphics cards. However, it does save a PCI slot, and when used properly, it has the potential of taking lots of traffic off of your PCI bus. See <www.tomshardware.com> for a discussion of this. The Diamond Steath 3D 2000 4MB PCI is a good lower-end board for $90.

CD-ROM

There are many good CD-ROM drives. For $100, I like the Toshiba SCSI XM5701B. If you don't have SCSI, get an ATAPI CD-ROM drive for about $60 to work on the IDE controller. <www.tdl.com/~netex> has a nice list of CD-ROMS.

You may wish to consider the DVD drives. Even though the standards are still in flux, the Sony 5X DVD-ROM drive for $388 is a good bet. Sony offers DVD+RW compatibility as a free option.

Floppy

These don't matter much. The TEAC 1.44 FLOPPY drive works and costs $25.

Ethernet

Today you may want to buy a 10/100 card even if your network is running at only 10Mb/s. We have measured the Intel Etherexpress Pro 100+ ($82) at 98Mb/s. It is based on the 82558 chip. Almost as fast, based on the 82557 chip, is the Intel Etherexpress PRO 100B at $60. Cards based on the DEC 21140 chip (de500ba) are also very fast and efficient. Stay away from the nameless cards.

Monitor

If you are spending a lot of time in front of the computer, get the best monitor you can afford. Once your eyes get bad, no amount of money will bring your eyesight back. A good monitor will substantially lower stress and fatigue – again, it's worth it. If you are working for someone, make them understand that you will be happier and more productive if you are looking at a sharper, clearer, no-flicker screen. The economics are compelling.

Monitors are always being rated in the PC magazines and trade rags. They provide much more detail than the following presentation, but for a superficial set of opinions, read on. And see <www.pcguide.com> for a description of monitor specs.

For X11 work, I consider a 17-inch monitor the minimum acceptable size. A 19-inch monitor is better, and you may want to go to a 21-inches for some applications. Note that you want a good 19-inch monitor more than a mediocre 21-inch version. To run a 1280x1024 screen on a 17-inch monitor, you will need dot pitch at least as fine as .25mm. If the dot pitch is larger, you won't be able to distinguish all 1280x1024 pixels. For vertical lines to be as sharp as horizontal lines, you'll need plenty of video bandwidth. Too little and your screen will flicker. I like the monitors shown in the table on the facing page.

Ken Merry's "theory" on monitors is that you want to run your monitor at one step below its maximum rated resolution. Thus, if the monitor is rated to 1600x1200, run it only at 1280x1024 and it will look great. See <www.tdl.com/~netex>, <www.pricewatch.com>, and <www.cdw.com> for details on monitors.

Manufacturer	Model	Size	Pitch	Maximum Resolution	Price
Nokia	445XPro	21"	.21mm	1800x1440@80Hz	$1,350
Nokia	446XPro	19"	.21mm	1600x1280@80Hz	$955
Nokia	447XPro	17"	.25AG	1600x1200@76Hz	$680
ViewSonic	G790	19"	.26mm	1600x1280	$785
ViewSonic	P775	17"	.25mm	1600x1280@76Hz	$550
ViewSonic	PT775	17"	.25AG	1600x1200@77Hz	$670
Sony	400PS	19"	.26AG	1600x1200@75Hz	$955
Sony	200PS	17"	.25AG	1280x1024@75Hz	$750
Sony	200GS	17"	.25AG	1280x1024@75Hz	$630

Keyboard

The KEYTRONIC 104KEY Lifetime Series PS/2 is solid, reliable, and cheap ($35). You may prefer the contour of the Microsoft Natural ($69).

Mouse

I recommend the Logitech MouseMan three-button version, which costs $25. For $45, you can get the new M-CV46, a contoured version with a forth thumb button. (Stay away from the ones with the scrolling wheels – they get in the way.) If you get stuck with a two-button mouse, most Xservers allow you to emulate a three-button mouse by pressing the two buttons at the same time.

Tape/Backup/Offline

Somewhere, somehow, you'd better be backing up your material! The main modest-cost tape drive contenders are 4mm DAT, 8mm, and DLT, in ascending order of price. These are all SCSI devices that are fast, reliable, and proven. (I look forward to Sony's new AIT, but for now it's unproven and quite expensive – $3,000 or so.) For under $1,000, you can get HP's C1554A, which is a DAT DDS-3 storing up to 12GB. The Exabyte 8mm and DLT drives are considerably more expensive. I recommend you stay away from weird controller boards, Travan, 1/4 inch, or other low-cost things. Visit <www.necx.com> for prices on various tape drives.

The 4mm DAT drives basically come in two flavors these days: DDS-2 and DDS-3. DDS-2 is 4GB uncompressed, 8GB compressed, with the 120m DDS-2 DAT tapes. You can get these tapes for $15 – $20. Backup speed is approximately 0.5MB/sec. DDS-3 is 12GB uncompressed, 24GB compressed, on a 125m DDS-3 tape. You'll pay about $25 per tape for these. Backup speed is roughly 1–1.5MB/sec.

If $1,000 is too much, consider getting a refurbished Conner/Seagate (Archive, actually) DDS-2 drive with 4GB/8GB for around $450 from Insight, <www.insight.com>. I've had a drive similar to this for quite a while, and it has worked great. If you can dig a little deeper into your budget, they might still have some refurbished Seagate (Archive Python) DDS-3 drives left for about $665. Don't just look at the specials they have listed; hit the specials search page and type in variations on "DDS," "DDS2," "DDS-2," "DDS3," etc.

You may want to consider alternatives to tape, such as the Iomega SCSI Zip drive ($150) with disks that cost $7. For more storage look at the Jazz 1GB or 2GB drives for

Integrators can save you lots of time and maybe some money because they buy and stock the components in bulk.

$450 and $90 per 1GB disk. These can also be used as hard drives. The writable CD drives are another option for backing up your system. The drives are down to $450 now, with blank disks costing only a couple of bucks each.

Printer

UNIX systems get along well with Postscript. An HP LaserJet 6MP makes an excellent, high-quality, medium-volume printer for $900. The Lexmark Optra S 1250 Laser Printer ($950) prints 12 pages per minute at 1200 x 1200 dpi. The 16ppm faster Lexmark Optra S 1620 costs about $1,000. The Lexmark Optra S 1620n for $1,300 comes with 10/100 network connections. The Lexmarks, like the HPs, handle PostScript and PCL. You can pay extra for the Ethernet connected printer, or just put it on a parallel port and use lpr software in your network. For not too much more money, you can get "color" laser printers that produce almost photo quality prints for about 35 cents each. Pages that have just a little color can be as cheap as 5 cents or so. See <www.qms.com>, <www.lexmark.com> and <www.hp.com>.

For much less money, you can get ink jet printers. The print quality has improved significantly over the years, but it is still inferior to laser printers. Color ink jet printers are also inexpensive. The HP Deskjet 500c is a good candidate, but unfortunately, most of the ink jet printers don't have Postscript. You'll need to install Ghostscript and APSfilter to convert all of your work into PCL. It is a workable solution if you need to save money and if you don't need laser printer quality.

Sound Cards

Sound cards and multimedia equipment in general are difficult areas to deal with on PCs running UNIX. Check drivers and recommendations on the appropriate OS homepage before you buy. Here are some other references: <www.opensound.com> and <www.iet.unipi.it/~luigi/FreeBSD.html>.

Universal Power Supply

I keep my systems up around the clock. Consider getting a UPS to help you around the power glitches and blackouts. You don't need to spend much to handle the one- or two-second interruptions. A simple system like an APC Back-UPS 400 for $130 will accomplish this.

Complete Systems

The prepackaged systems such as those from Micron, Gateway, Compaq, HP, and Dell represent good value in the WinTel world. These systems have been optimized for price and compatibility with Windows 95. You'll find they don't all work well with NT and/or UNIX. Generally, the stumbling block for UNIX is the video/graphics card. If you find a prepackaged system with one of the supported cards, you'll probably be fine.

Keep in mind that to keep the price down, they generally use less reliable power supplies, cheap IDE drives, and other off brand components such as modems and graphic cards. These often cause headaches, negating the cost advantage. If you need to buy a bunch of boxes, you may be able to amortize the cost of figuring out these issues; for the one- or two-system situation, save yourself the trouble by buying what is known to work well.

Integrator

Integrators can save you lots of time and maybe some money because they buy and stock the components in bulk. Here you can get a system built from the components you specify. Ask one of the following vendors for a quote on the system you want. Also,

visit their Web sites and look at the details of their prepackaged systems. Maybe they already built what you want. Some of the vendors will even preload your source code UNIX system. Also see <www.linux.org/hardware/systems.html> for other resellers.

<www.tesys.com>
<www.tdl.com/~netex>
<www.asacomputers.com>
<www.apache.com>
<www.varesearch.com>

Summary

For a low system, look to save money with a Pentium/MMX, AMD, or Cyrix processor. Live with IDE drives: both hard disk and CD-ROM. You'll be using plain, cheap EDO or FPM DRAM. Get at least 32MB – more will help, but insist on parity detection. You can save some money with a lower cost video/graphics card. You might drop down a notch or so with a cheap power supply and case, but if possible, stick with the ATX form factor. Watch for the clear-out advertising. For example, OfficeMax is dumping Pentium 166 systems for $700 and 233 systems for $900. But be careful about the video card. Or build your own based on Socket-7 CPUs. Check out <www.surplusdirect.com> for deals.

For the medium system, you'll want a more current processor, such as Pentium II or K6. Get the fast SDRAM to go with it. (i.e., FIC motherboard for K6). Spend money on SCSI drives. Upgrade to a Matrox Millenium II graphics card. Here are the components of a medium system I'm building:

$ 300	INTEL 233 MHz Pentium-II, 512K cache
275	ASUS P2L97-S motherboard, 4PCI/2ISA, AGP, SCSI
250	MEMORY ECC 8X72 64MB-10NS, SDRAM
400	SEAGATE SCSI 4GB ST32272N, 7200
210	Matrox Millenium II 4MB
24	TEAC 1.44MB FLOPPY DRIVE 3.5"
25	KEYTRONIC 104KEY, W95, PS/2(ATX)
21	LOGITECH MOUSEMAN 3BT PS/2
90	INWIN MIDTOWER/ATX, IW-A500, 8BAY, W/230WPS
24	CPU COOLING FAN, PC-POWER & COOLING, P5-II, 5
10	COOLING FAN, 8X8CM(3.14")-ADDITIONAL
84	TOSHIBA SCSI 12X XM-5701B
80	NET-Intel INTEL PRO 100B ETHERNETCARD
4	SCSI STD-CABLE INTERNAL/RIBBON, 3-CONN
$1,797	

For the high system, get a motherboard based on the 440BX or 450NX that runs the system bus at 100MHz. Get a Pentium II 350/100/512 or 400/100/512 or even the Pentium Deschutes processor, and think about multiple processors.

Sources and Reviewers

I am indebted to the following engineers: Ken Merry, Michael Durian, Tom Poindexter, and Joel Rem. They know and understand hardware; I just tried to summarize their knowledge.

java performance

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

This column is the first in a series on Java performance. Some of the discussion will be on features unique to Java, some on general performance issues in a Java setting.

Method Call Overhead

In a language such as C or C++, you might have encountered the idea that calling a function ("method" in Java) has some time overhead associated with it beyond the actual processing done by the function. For example, with this code:

```
int max(int a, int b)
{
  return (a > b ? a : b);
}

int main()
{
  int i = 0;
  int j = 0;
  int k = 0;
  long n = 500000000L;
  while (n-- > 0)
    //k = max(i, j);
    k = (i > j ? i : j);

  return 0;
}
```

coding the actual calculation of the maximum of two ints is about three times as fast when done inline as when done via a call to `max()`. In C++ there is an "inline" specifier that can be used to specify that a function should be expanded inline if possible.

Function call overhead is caused by various factors, including time spent in setting up stack frames, actual transfer of control to the called function, and so on.

Java also has overhead with calling methods. For example, in this code:

```
public class test1 {
  public static int max(int a, int b, int c)
  {
    return (a > b ?
        (a > c ? a : c) : (b > c ? b : c));
  }

  public static void f(int a, int b, int c)
  {
    int m = 0;
    int n = 500000000;

    while (n-- > 0)
        //m = max(a, b, c);
        m = (a > b ?
          (a > c ? a : c) : (b > c ? b : c));
    }

    public static void main(String args[])
    {
        f(37, 47, 57);
    }
  }
```

doing the max computation inline is about twice as fast as calling the method `max()`.

In Java, one reason for overhead is that methods are virtual by default, that is, the actual method called is determined at runtime by considering the actual object type. For example, the Object class, from which all other class types extend, has a method `hashCode()` defined in it. A class that extends from Object may also have a `hashCode()` method. If in a program you have an Object reference and wish to call `hashCode()`:

```
public void f(Object p)
{
  int h = p.hashCode();
}
```

then the actual version of `hashCode()` called depends on whether `p` is "really" an Object or refers to an instance of a class derived from Object.

So, because a method is virtual by default, it is difficult or impossible to expand it as inline (in C++, some clever optimizations have been used to make virtual functions inline, but this is a hard problem).

The keyword "final" can be used to say "this method cannot be overridden" by another method in a derived class. Note that "final" has various other meanings, for example saying:

```
final int N = 10;
```

marks N as unchangeable, and a final class cannot be extended from. Such a declaration is a hint to the compiler, nothing more, and there's no guarantee that a given method will be expanded inline.

Should you worry about method call overhead? Most of the time, no. It's only when you have a method that's called very heavily in a loop that this overhead matters.

Final Classes

In Java, a class organization such as:

```
class A {}
class B extends A {}
```

results in a superclass (A) and a subclass (B). References to B objects may be assigned to A references, and if an A reference "really" refers to a B, then B's methods will be called in preference to A's. All of this is a standard part of the object-oriented programming paradigm offered by Java.

But there is a way to modify this type of organization, by declaring a class to be final. If I say:

```
final class A {}
```

then that means that A cannot be further extended or subclassed.

This feature has a couple of big implications. One is that it allows control over a class, so that no one can subclass the class and possibly introduce anomalous behavior. For example, `java.lang.String` is a final class. This means, for example, that I can't subclass String and provide my own `length()` method that does something very different from returning the string length.

There is also a big performance issue with final classes. If a class is final, then all its methods are implicitly final as well, that is, the method is guaranteed not be overridden in any subclass. A Java compiler may be able to inline a final method. For example, this program:

There is also a big performance issue with final classes.

```
final class A {
  protected int type;
  public int getType() {return type;}
}
public class test2 {
  public static void main(String args[])
  {
   int N = 500000000;
   int i = N;
   int t = 0;
   A aref = new A();
   while (i-- > 0)
    t = aref.getType();
  }
}
```

runs about three times as fast when the class is declared final.

Of course, much of the time it's desirable to use the superclass/subclass paradigm to the full and not worry about wringing out the last bit of speed. But sometimes you have heavily used methods that you'd like to have expanded inline, and a final class is one way of achieving that.

using java

Using Beans within ActiveX

by Prithvi Rao

Prithvi Rao is the founder of Kiwilabs, which specializes in software engineering methodology and Java training. He has worked on the development of the MACH OS and a real-time version of MACH, and he holds two patents resulting from his work on mobile robots.

<prithvi@kiwilabs.com>

We live in a world of diversity, and the computing industry is no exception. Recently, I was writing some ActiveX code just to learn more about it, and I had an epiphany: Java Beans and ActiveX are remarkably similar.

My first thought was, "Why should I bother learning this technology?" My second was, "maybe I shouldn't." I am not suggesting that ActiveX is either a lesser or a redundant technology, but that it is an alternative to Java Beans. It also does not mean that if you write code in Java and it acts like a Bean, you are excluded from ever incorporating your Beans as reusable components into an ActiveX environment.

The focus of this article is precisely how to do just that. I am going to go through the steps of taking a piece of code written as a Java Bean and embed the Bean into an ActiveX control. I am going to look at the ActiveX bridge provided with the Beans Development Kit from Sun (BDK 1.0, <http://java.sun.com>).

The reasons for doing this are quite compelling. ActiveX and Beans are the two main component architectures. ActiveX, which is based on the Component Object Model (COM) from Microsoft, first made its debut in 1992. Recently, Sun came out with its component architecture, Java Beans, and it has gained a great deal of popularity since its inception in 1997. Java Beans are written entirely in Java and require a JVM. The Java Development Kit is available from <http://www.javasoft.com>.

I am not going to deal with the ActiveX side of things except where it is necessary to understand the process.

Beans and ActiveX

The following summarizes the two component architectures:

Beans	ActiveX
Written in Java	Written in C, C++, or VB
Requires a JVM	Requires COM
!Pointers	Pointers
Simple to use	Complex to use

My experiences and those of my colleagues lead me to the opinion that it is relatively difficult to learn ActiveX in a short time and be productive. However, if you have a large investment in ActiveX code and all your new and recent hires (who all happen to Java devotees) are going to be developing components using Beans that will work with legacy ActiveX code, then read on.

Beans Defined

A Java program is a Bean under the following conditions:

1. Its private data fields are available using accessor methods.
2. Accessor methods are public methods.
3. Standard design patterns are used to name private data fields.
4. It uses introspection or a BeanInfo class to get information about private data fields.
5. It can be used with property editors.
6. It can be serialized to provide persistence.
7. It uses a delegation model for handling events.

The ActiveX container and the Bean never communicate directly; everything goes through the bridge.

Beans expose properties, events, and methods; this is identical to ActiveX. Regardless of how an ActiveX control was written, it won't know that it is interacting with a Bean if we can find a way to expose the properties, events, and methods of the Bean to the ActiveX control. This is where the ActiveX bridge comes in.

The ActiveX container and the Bean never communicate directly; everything goes through the bridge. Currently, the bridge is in its third beta release.

Let's create a visible Bean (the second beta release supported only visible Beans) to see how to "bridge" the gap.

Writing the Bean

The following Bean is from Rob Englander's book, *Developing Java Beans*, and I use it with his permission. We are going to look at code for the `BeansBook.Simulator.VisualTemperature` class. Let's call this file `Simulator.java` and put it in `~/projects/MyBeans/Java`.

```
package BeansBook.Simulator

import java.beans.*;
import java.awt.*;

// let's define the temperature bean class
public class VisualTemperature extends Panel
  implements TemperaturePulseListener //detector listener
   java.io.Serializable // make it serializable
{

// provide support for bound property listeners
protected PropertyChangeSupport boundSupport;

// the current temperature
protected double theTemperature = 22.0;

// constructor for this class
public VisualTemperature()
{
  // construct the support object
  boundSupport = new PropertyChangeSupport(this);
}
  // add a property change listener
  public void addPropertyChangeListener(PropertyChangeListener)
  {
   // defer to the support object
   boundSupport.addPropertyChangeListener(l);
}

// remove a property change listener
public void removePropertyChangeListener(PropertyChangeListener l)
{

  // defer to support object
  boundSupport.removePropertyChangeListener(l);
}

  // get the value of the Temperature property
public double getTemperature()
{
  return theTemperature;
}

  // set the value of the Temperature property
public void setTemperature(double t)
{
```

```
    // do nothing if the value did not change
     if (t == theTemperature)
     {
       return;
     }
    // save the old value
     Double old = new Double(thetemperature);
     // save the new value
     theTemperature = t;
     // fire the property change event
     boundSupport.firePropertyChange("Temperature", old new
                                          Double(t)); }
    // handle a temperature pulse
    public void temperaturePulse(TemperaturePulseEvent evt)
  {
     // get the pulse temperature
     double p = evt.getPulseTemperature();
     // get the current temp
     double c = getTemperature();
     // if the pulse temp is > current temp
     if (p > c)
     {
      // only change if the difference is > 1
      if ((p - c) >= 1.0)
      {
       // add 1 to the current temperature
       setTemperature(c + 1.0);
      }
     }
    }
    // take a temperature pulse by a direct method call
    public void directTemperaturePulse(double pulse)
    {
     // emulate a regular temperature pulse
     temperaturePulse(new TemperaturePulseEvent (null, pulse));
    }
  }
```

This Bean conforms to standard design patterns for accessor methods, handling events, and so forth. I'll discuss design patterns in more detail in subsequent articles.

Jarring the Bean

All Beans are "packaged" as Java ARchive files (JAR). If we want to embed the Bean in an ActiveX control, we need to create this archive file. I'll call it `Simulator.jar`.

The first thing to do is to compile the `Simulator.java` file:

```
javac Simulator.java
```

This creates a class file called `Simulator.class`

Next, create a JAR file by using the `jar` utility that is part of the JDK. Go to the directory containing the `Simulator.class` file, and run the `jar` utility:

```
jar vcf Simulator.class Simulator.jar
```

This command is a directive to create a new archive called `Simulator.jar` and make the file `Simulator.class` be part of it. The `v` switch provides verbose output and is not absolutely necessary. You might have observed that `jar` is very similar to the `tar` utility. `Jar` essentially compresses the file. This is especially important because Beans are intended to be accessed across a network.

The popularity of Java, and therefore Beans, is increasing, but the ability for Beans to interoperate with other component architectures will partially determine its success.

Packaging the Bean

This is the main step that actually makes the Java Bean recognizable by the ActiveX control.

First, create a directory where you can place all the output from the Packager. This is where all the type libraries and registry files created will be placed, as well as the Java stub files. All of these are outputs of the Packager.

```
mkdir ~/projects/MyBeans/ActiveX
```

Now copy the `beans.ocx` file in `bdk/bridge/classes` into this directory. This is the copy of the `beans.ocx` that will be referenced by the ActiveX versions of the simulator Beans.

Before we run the Packager, we need to set the `CLASSPATH` environment:

```
setenv CLASSPATH ~/bdk/bridge/classes
```

Now we can run the Packager:

```
java sun.beans.ole.Packager
```

The Packager presents you with a dialog box that walks you through the steps needed to package a Bean. Here are the steps:

1. Specify the JAR file previously created containing the Bean. You can type in the full pathname of the file

   `~/projects/MyBeans/Java/Simulator.jar`

2. The dialog box changes to show the Beans contained in the JAR file. Select the class "BeansBook.Simulator.Thermostat," and press the "Next" button.

3. Now you have to specify the ActiveX name that should be used to identify the component. Let's use "JavaThermostat," and press the "Next" button.

4. Now specify the directory into which the type library and the registry files are to be placed. These must be in the same directory as the `beans.ocx` file. (Remember we copied it into `~/projects/MyBeans/ActiveX`.)

5. Next we are asked if we wish to "crack" the events. The discussion on this topic is for another time. For now let's "crack" the parameters. This means that event is fired into the ActiveX event firing protocol.

6. Now start the packaging process by pressing the "Start Generation" button.

The Packager tool adds new classes to the JAR file, and these classes all start with `ole`, which stands for "Object Linking and Embedding." One new class supports the Bean's ActiveX control interface. The rest are to support the ActiveX control event listeners.

If you wanted to use the Bean in a Microsoft Word document, you would use the `Object` command to select this control. The Java Bean can now be used anywhere an ActiveX component can be placed.

Conclusion

Understanding interoperability is a fundamental requirement of the computing world because it is one way to bridge the gap between newer and older technologies. Examples of this are CORBA/DCOM and JDBC/ODBC. The popularity of Java, and therefore Beans, is increasing, but the ability for Beans to interoperate with other component architectures will partially determine its success. Java Beans encourages the programmer to live in the design space of a problem. With the powerful, easy-to-use tools provided by the BDK distribution, we can make interoperability a reality.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of UNIX System Security and System Administrator's Guide to System V.

<rik@spirit.com>

One of the fun things about being an editor, as I was for the security special edition of *;login:*, is that you get to read everything before anyone else does – read it not only once, but perhaps several times as the articles move through the production process. So it was that I had the chance to read Dan Geer's thoughtful and erudite speech about the future of the security market several times. I wrote Dan that I didn't agree with all of his points, and he responded by offering to debate them. I declined, but was still troubled by several things he said.

One was that the large vendors are in the best position to provide security products. Vendors already sell service and support and have access to the software, so they are in a great position to provide vertical security integration. My thought was, well, so what? They have had this access for years without doing anything about it. Although the resolution of risk will be a driving force in the near-term future of computing, as Dan drove home, I have seen vendors selling a firewall as a "service" that cost $150,000 to set up. At USENIX, a sysadmin from Sun presented a paper in which she described internal software for closing all the known security holes in Solaris. The internal software was not available outside of Sun, and one wonders why Sun didn't just patch its distributions. With a track record like this, I wonder what the big vendors will do with their opportunity to provide risk management.

The Commons

The statement that really stuck in my mind was something else. Dan mentioned Garret Hardin's term "tragedy of the commons." Hardin's observation was that what is shared in common tends to be subdivided and vandalized over time. The commons referred to were public lands that townspeople could use "in common" for purposes such as cattle grazing. But my wife suggested a more recent example: dogs being banned from a large playing field because their owners didn't clean up the piles of doggy poop left behind. This is a good example of a subset of people ruining a common space for all through their "vandalism."

Dan was referring to the future of UNIX, not to fields. The fate of UNIX as an open system has been that each vendor that licensed it changed it sufficiently to make software porting a big issue and system administration of heterogeneous systems a giant pain. And the various versions of UNIX continue to diverge further, fragmenting what might have been a single interface. When people ask me what I think about the future of UNIX, I tell them that UNIX is not going away, but that other operating systems will replace it in the future.

I see the people who work on the various flavors of BSD and on Linux as those who will be writing the operating systems of tomorrow. They will have firsthand, in-depth experience modifying and improving operating systems. They will also have had the experience of working publicly, in common, with their code exposed for review and criticism by all so motivated. A tough school indeed. And the operating systems they so tenderly nurse are being used today.

Greed

What of the commercial OS vendors – in particular, a large vendor that many have grown to fear and loathe? Having just read Dan's speech yet again, I came across an article about economics and sociology in *Science News* (Bruce Bower, "Yours, Mine and

Looking into my famous crystal ball, I see, instead of an enormous operating system that does everything and has a totally proprietary and closed API, an API that is not owned by a single company.

Ours," vol. 153, pp. 205–207). Reading this article allowed me to view the commons in a different light, one I believe will influence the business of computing.

Any of us who know anything about economics will readily agree with Bower's assertion that free markets establish prices based upon rational decisions by the buyer and the seller. Questions of ethics and morality have no place in these decisions as each player tries to slice off as big a piece of the pie as possible.

However, Bower's article is not about ideal markets, but rather about the people who participate in the decisions that define those markets. People, you and I, live in a cultural matrix, including offices, churches, professional organizations, unions, and neighborhoods; and our matrix influences our decisions. In particular, we tend to punish those who appear too greedy, even if it means we too will suffer.

Bower describes several experiments in which the results vary, depending on the culture in which the participants live. For example, there is the "ultimatum game," where one player is given a large sum of money but must share it with the other player. What makes the ultimatum game interesting is that the other player must find the first player's proposal acceptable or neither of them gets any money. In some cultures, the proposer offers an even split, especially if the amount of money is large (three month's wages). In Western cultures, the proposer typically offers between 30 and 40%, and offers below 20% are generally refused. Putting this in real terms, if the proposer has been given $500 and offers only $100 of it to the other player, the other player (who could have had $100 by simply saying yes) refuses, and neither gets any money.

Whoops! This isn't rational at all. The second player is punishing the first for being too greedy. This behavior, the willingness to give up rewards simply to punish a greedy player, shows up in other games as well. In real life, this shows up in some peoples' attitudes toward Social Security (acceptable, because the recipients had worked) and welfare (not acceptable because the recipients did not participate or add to the common good).

What does this have to do with operating systems? Imagine OS vendors playing the ultimatum game with each other (and with us, their customers), but instead of offering money, they offer us OS features. These features include networking, high performance, ease of maintenance, and support for applications, among other things. What has made Microsoft so successful has been its support for applications – more applications, and more users of those applications, than any other OS platform.

But herein lies the problem. The interface used by those applications is itself not open, not shared. As the importance of the interface (the API) grows, while Microsoft moves constantly to defend and protect that interface so that it is not shared, the perception of a proposer who is not playing fair also grows. As a society, we have tended to punish those who got too greedy (remember Michael Milken?) or abused their power (Richard Nixon). And Microsoft is definitely not playing "fair."

Looking into my famous crystal ball, I see, instead of an enormous operating system that does everything and has a totally proprietary and closed API, an API that is not owned by a single company. The API provides a specification that can be supported by many operating systems and will cleanly support extensions to the operating systems themselves. For example, the addition of new devices or networking protocols would be supported by this API. The point of an open API is that there can be many players. Or, in terms of the ultimatum game, the organization offering the API is proposing an even sharing, which will not stimulate a punishing backlash against greedy behavior.

Until I mentioned devices and networking, you might have thought I was talking about POSIX, but I'm not. Modern operating systems have requirements beyond those of POSIX, and any API that does not include GUI elements is doomed to failure. Just as POSIX 1 was based on the C interface to UNIX, a new API might be based on some popular computer language. Examples include Perl, Tcl/Tk, and Java, because each of these languages has a core that remains the same on all platforms.

Java may already be sunk too deep into intrigue to become a truly open standard. HP recently, with Microsoft's encouragement, published its own Java standard for embedded devices. If Sun were to bust JavaSoft loose, then perhaps Java would have a chance. John Ousterhout, the author of Tcl/Tk, has left Sun, giving Tcl/Tk a decent chance, although Perl today is vastly more popular. The problem with Perl is that it may be just too damn flexible for describing an interface.

I don't want to get into religious wars about which language is best. I'd rather think about reintroducing the commons in terms of an API that will support the types of applications we are writing today and will write in the future. A common API (with room for some well-mannered, portable extensions) will permit vendors and writers of operating systems to compete on an equal basis. People who talk about Microsoft OS being superior "because the market has chosen" might understand markets, but not software.

A common API (with room for some well-mannered, portable extensions) will permit vendors and writers of operating systems to compete on an equal basis.

Just before I sat down to finish this, I saw an IBM ad about Web security and thought of Dan again. The trouble with IBM, or any other large vendor, as the source of security is that its solutions won't support the commons. Any proprietary security solution will fail, because today's systems must interoperate. The very idea of IBM Web security being available as part of an operating system sold by Microsoft, Sun, HP, or DEC is still ludicrous. Security must be like networking, truly interoperable, if it is to succeed.

I'd like to end on a more personal note. I enjoy being part of a community. I find there is pleasure in becoming part of something larger, whether it is a pickup volleyball game in Golden Gate Park, USENIX, or sharing knowledge. I believe lots of things separate us today, many artificial boundaries, and I think that we would all feel better without those boundaries.

faq-o-matic

by Jon Howell

Jon Howell is a graduate student working with David Kotz at Dartmouth College. His research is on single-system-image operating environments that span adminstrative domains. He also enjoys robotics, web glue, and kernel tweaking.

<jonh@cs.dartmouth.edu>

What is the best way to maintain a Frequently Asked Questions list? Traditional FAQs are maintained by hand and often become outdated as the maintainer loses interest in the task. Mailing list archives can be helpful, but are often too disorganized. When I found myself answering the same FAQs about Linux for PowerPC and MkLinux last year, I faced this very problem. I am far too lazy to commit to maintaining a FAQ, but the mailing list archives were not significantly reducing the load of redundant questions.

Solution Design

The FAQ-O-Matic is a Web-editable FAQ designed to offer a middle ground. Because the general public can edit it, it is less likely to become neglected and stale like a manually maintained FAQ. Because changes are submitted where the FAQ is read, one can be rather lazy and still contribute to the FAQ. No one person has the responsibility of maintaining the entire document.

Because a FAQ-O-Matic is topically and hierarchically organized, browsing solutions is easier than it is in mailing list archives. Queries on mailing list archives can return matches for old, outdated information and miss newer answers. The topical organization of a FAQ-O-Matic helps avoid this problem as well.

A search function makes FAQ-O-Matic as accessible as a mailing list archive. Another function lists recently changed items, so users can check back for changes if they did not find a satisfying answer the first time they looked. There is a command to show entire categories in one HTML file, to facilitate printing or export of the FAQ.

How It Works in Practice

I launched the first FAQ-O-Matic by seeding it with about 60 or 70 answers gleaned from recent list postings. Although this opposed my laziness philosophy, I knew that I would not be responsible for keeping the answers up to date. Then I began advertising it by answering questions with URLs to the FAQ-O-Matic.

One problem with the initial implementation was that answers were identified by their location in the topic hierarchy. So if you sent out a URL to a FAQ-O-Matic answer and the database was subsequently reorganized, that URL would go sour.

I initially thought allowing people to submit questions without answers would help define the structure of the FAQ by reserving spaces for answers when they became available. Instead, people who were too lazy to search would post poorly considered questions in inappropriate categories.

The submission page asked users to leave an email address with their comment, but people often forgot or inserted text between previous text and its attribution. Furthermore, although the server uses RCS to protect against wholesale vandalism, there was no way to trace subtle, intentional corruption of the database.

The FAQ-O-Matic allowed the entry of HTML tags, so users could supply links and formatting information for their answers. However, other than links, HTML was rarely used to good effect. Instead, it often made for inconsistent appearance as people appended to existing answers and as HTML tags fought with the formatting generated by the CGI script. Furthermore, code segments pasted into the FAQ-O-Matic would mysteriously lose < and & symbols.

Finally, I found that I had to put in a certain amount of effort moving answers around to keep them organized as new answers showed up. This was compounded by the diffi-

culty of performing this sort of administration on the first implementation of FAQ-O-Matic.

Version 2

Over the summer, I rewrote the FAQ-O-Matic to address these problems. First, each answer is now assigned a permanent ID number. This solves the sour URL problem and also provides a facility for "see also" links inside the FAQ.

I posted a policy disallowing questions without answers, which trivially solved the second problem.

The new version has an authentication scheme that verifies email addresses by sending a secret validation code to the given address. Thus each submission is attributed to a real email address, and intentional corruption, once noticed, can be traced and rolled back.

FAQ-O-Matic 2 no longer allows HTML tags; they are displayed as entities (<). This prevents code from becoming corrupted and enforces a uniform (if bland) appearance. Links are supported heuristically by detecting things that look like links (http://...) and generating HTML links on the fly. Internal links are supported with a special URL that begins "faqomatic:"

Version 2 also has support for reorganizing answers and categories from the Web interface. This facility might allow a Web user to moderate a section of the FAQ and care for its organization. Moderators can request that they receive mail whenever any answer in their area is changed, minimizing the effort associated with the moderation task.

The 1998 LinuxPPC CD was announced in January, prompting an estimated 4,500 people to visit the FAQ-O-Matic on <www.dartmouth.edu> in one day. Because every request required service by a Perl CGI, the memory pressure overloaded the server, and the FAQ-O-Matic had to be throttled. In response to that event, version 2.6 adds a server-side HTML cache, so that people who are only reading the FAQ receive HTML directly from the Web server, without the cost of invoking the CGI.

Other Uses

Because FAQ-O-Matic has an authentication scheme, it made sense to give it flexible authorization as well. The FAQ can be configured to be very open, not even requiring mail-back email secrets, to encourage the shy, lazy, or anonymous contributor at the expense of accuracy of attributions.

Alternatively, it can be set to allow only assigned moderators to modify the FAQ. In this arrangement, it is suitable for use as a help desk database: only help desk operators can modify the database, but it is available for all to read.

Numbers

The Linux on PowerPC FAQ-O-Matic has been available for 15 months. In that time, about 75,000 different people (IP addresses) have seen it, and it has received 1,500 submissions. On average, visitors access it about ten times. A few dozen people claim to be running their own FAQ-O-Matics, some for internal projects, others for Internet-accessible sites.

Conclusion

The FAQ-O-Matic has turned out to be a successful system for documenting the Linux on PowerPC projects. It is more organized than a mailing list archive, but avoids going stale as traditional FAQs often do. It allows a division of labor in maintaining both answers and the overall organization of the FAQ. And it has access control features that make it suitable for other applications. To try it out, visit the FAQ-O-Matic at <http://www.dartmouth.edu/cgi-bin/cgiwrap/jonh/faq.pl>.

Because FAQ-O-Matic has an authentication scheme, it made sense to give it flexible authorization as well.

NT scalability III

How to Stack Cyberbricks

by Neil Gunther

Neil Gunther is founder and principal consultant for Performance Dynamics Company in Mountain View, CA. Dr. Gunther has worked in the Silicon Valley for 18 years. He is a member of IEEE, ACM, and CMG.

<ngunther@ricochet.net>

This is the third and final article in the series [1, 2] on NT and UNIX server scalability. The motivation stems from my concerns about the misleading comparisons between UNIX and NT scalability that were presented at the USENIX-NT Workshop last August. My final article has to do with defining what is actually meant by the term "scalability."

In his opening address [3], Gray never actually defined what he meant by scalability, but one wrong impression revolved around Microsoft's cluster strategy. The notion is to connect cheap PCs in the same way one combines bricks to build a wall; in this case the PC units are to be thought of as cyberbricks that can be combined via clustering technology into larger scale computational structures. Sounds good, doesn't it!? So where do you get yours? You can't, if you want the version Gray described, because it's not ready for prime time (recall that the two-node cluster demo failed to appear during Gray's presentation [1]). Pyramid, DEC, Sun, HP, et al., however, do have UNIX clusters on the market and in development.

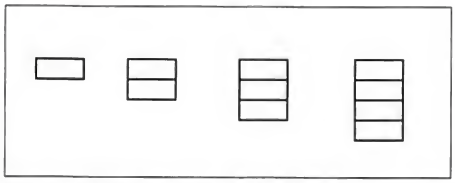


Figure 1. Ideal stacking. This doesn't account for cementing the Cyberbricks together.

Let's examine this scalability concept more closely. In Figure 1, the area bounded inside each box (or "brick") represents the maximum available PC computational capacity.

If we include the "cement" or "glue" needed to build a larger computational structure with these Cyberbricks, we end up with the situation shown schematically in Figure 2. The "cement" in this case is the overhead cycles consumed at each PC node (due to both hardware and software) as they communicate with one another. Let's see how this would work in more detail.

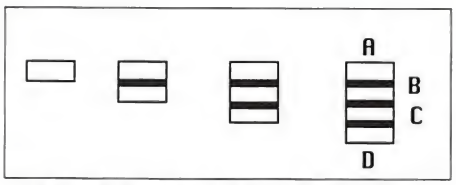


Figure 2. Cementing of up to 4 Cyberbricks

Adding Cyber-Cement

As with real bricks and mortar, we have to prepare the surface of each brick where the mortar will be applied to stick the bricks together. Let the percentage of each Cyberbrick given up to the interface be g% of the available nodal computational capacity (i.e., g% of each Cyberbrick's volume for each interface). This lost fraction of capacity is shown as the black area between the bricks in Figure 2.

Although this is how things would look if these were real bricks, it turns out to be rather unrealistic for Cyberbricks. The lack of reality can be seen on both theoretical and empirical grounds.

Look at the stack of four Cyberbricks. It's clear that node A can interact with node B because they are "cemented" together. But how does node A interact with node C or node D? In this naive model of scalability, they simply can't! The effect of this limitation is surprising. To see this, let's take some concrete values.

Table 1. Naive scalability estimates

Cyberbricks	Faces	Loss	Difference	Capacity
1	0	0	1 – 0	1.00
2	2	2 x 0.05	2 – 0.1	1.90
3	4	4 x 0.05	3 – 0.2	2.80
4	6	6 x 0.05	4 – 0.3	3.70

If we assume that g = 5%, then we can estimate the available computational capacity as we scale up the number of physical nodes (p). For p ranging from one to four nodes, the results are summarized in Table 1.

Clearly, there is an impact. For example, when we add the second node, we don't get double the computational capacity, only 1.9 nodes of effective capacity (the discount being 5%, as expected). At 4 nodes, we get only 3.7 nodes of effective capacity. The full impact up to 8 nodes is shown in Figure 3.

The 5% discount (due to the chosen g-factor) applies across the board and simply means that the ideal scaling curve has a slightly lower slant (lower by 5%). So adding in the Cyber-cement like stacking real bricks (Figure 2) is more realistic than not including it all (Figure 1) but it is still an unrealistic model of real computers for two reasons:

1. It doesn't account for arbitrary internode interaction (e.g., A to D).
2. Interconnect latencies and software contention are not included.

If we attempt to include these more realistic effects, we end up with a slightly more subtle cementing procedure (Figure 4).

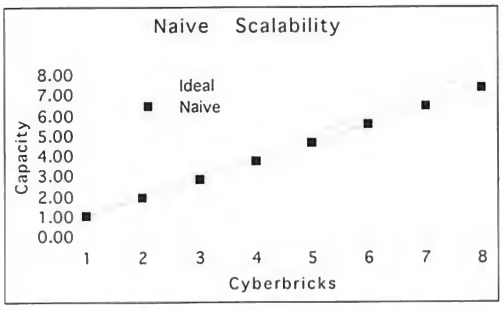


Figure 3. Naive scaling model up to 8 nodes or Cyberbricks

Taking the Edge Off

Cementing together two Cyberbricks is the same as before (Figure 2) so the effective capacity loss should be the same. To cement three Cyberbricks, we have to combine them in such a way that each Cyberbrick touches every other Cyberbrick. The third diagram in Figure 4 shows how to do this.

There are three segments of cement and two Cyberbrick faces for each segment, for a total of six faces that lose g=5% of each Cyberbrick's volume (i.e., a total loss of 6 x 0.05 = 0.3 nodes). Subtracting this loss from the 3 physical Cyberbricks leaves an effective capacity of 2.7 Cyberbricks. This is smaller than the previous case (Table 1).

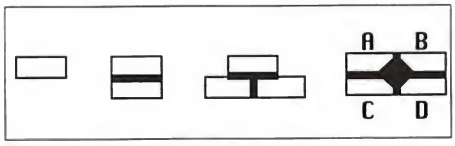


Figure 4. More realistic Cyberbricks configurations

When it comes to cementing four Cyberbricks, things get a little trickier. You might expect the stack to be like that shown in Fig. 5, but that configuration does not allow node A to interact directly with node D. The correct stacking arrangement is shown in the last diagram in Figure 4. The secret is that we need to take some additional capacity by preparing a bevelled cement face between A-D and B-C.

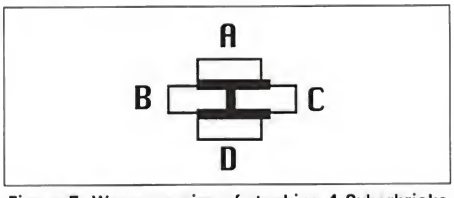


Figure 5. Wrong version of stacking 4 Cyberbricks

The total loss is assessed as follows:

4 x (2 x 0.05) for each flush face

2 x (2 x 0.05) for each bevelled face

Subtracting this combined loss from the total number of Cyberbricks, produces an effective capacity of:

$$4 - (4 \times 0.10) - (2 \times 0.10) = 4 - 0.6 = 3.40$$

For p = 1 to 4 nodes, the complete results are summarized in Table 2.

The full impact of this stacking procedure up to eight nodes is shown in Figure 6. The key difference from the naive scaling model (Figure 3) is the nonlinear characteristic in the scaling curve (very typical of real systems [4, 5]). This "curvature" in the scaling function reflects the increasing overhead of the additional communication paths as more Cyberbricks are added to the configuration.

Table 2. Realistic scalability estimates

Cyberbricks	Faces	Loss	Difference	Capacity
1	0	0	1 – 0	1.00
2	2	2 x 0.05	2 – 0.1	1.90
3	6	6 x 0.05	3 – 0.3	2.70
4	12	12 x 0.05	4 – 0.6	3.40

If you don't know the g-factor value, you can get it from measured throughput data (e.g., transaction per hour) across a few configurations. This can often be obtained from application developers doing pilot scalability studies or systems analysts who have been tracking upgrades on production systems. These data are entered into the array called "data" in the program called `gfit.c`. (See the source code on pp. 53-54.)

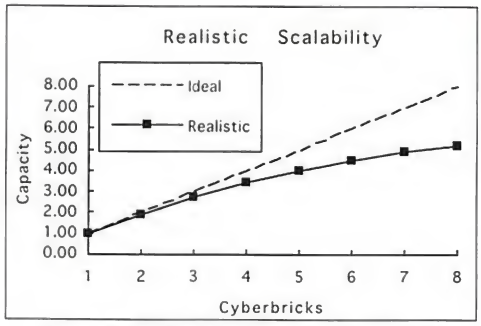


Figure 6: Realistic scaling model up to eight Cyberbricks

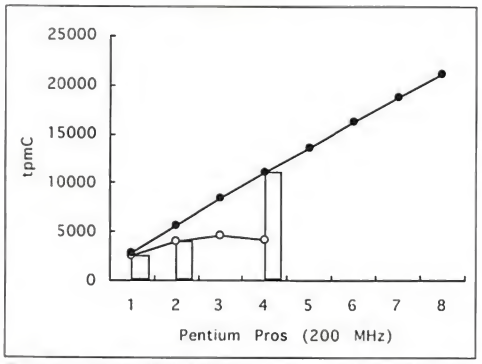


Figure 7: Modelling estimates for Pentium/SQLServer TPC data

Applying the Stacking Model

Let's see how this kind of simple model can be applied to estimating the scalability of real computer systems. Because I discussed TPC-C benchmark data previously, [2] I'll use that same source. Unfortunately, there are no TPC-C data for different cluster configurations (yet). In lieu of that, I'll apply the model to multiprocessor data based on the 200MHz Pentium Pro running Microsoft SQLServer (bars in Figure 7). This is permitted because the realistic stacking model I've presented is quite generic and does not assume any particular interconnect technology. A shared-memory bus (like the Intel P6 bus) is also accounted for by our model.

Turning to Figure 7, let's begin with the p=1 and p=2 CPU configurations. The respective TPC-C throughput numbers were reported as 2,502 tpmC and 4,939 tpmC. Applying `gfit.c` to these values, we get a g-factor of almost 20% (a very poor value, by the way!). Using this g-factor, we can project the throughput of a 4-way platform (white dots in Fig. 7). The estimate is 4,219 transactions per minute – worse than the two-way configuration! That's right. Losing 20% of your cycles for every communication path between CPUs implies that adding the fourth CPU is a waste of money (even adding a third CPU is questionable!).

But looking up the reported TPC-C result for a four-way, we find a remarkable result: 11,055 tpmC – very much better than expected (by almost a factor of three times!). Does this measurement invalidate our simple scaling model? No, it doesn't. There were many variables changed during the period (May–October 1997) between reporting the two-way TPC-C and the four-way TPC-C results (e.g., version changes in the O/S, the RDBMS, and the L2 cache size was doubled for the four-way platform).

Regardless of which of these changes had the biggest impact, we can use our scaling model to do some interpolation estimates. Let's suppose that 1MB L2 caches were also available on the two-way and single CPU configurations. If the g-factor can be reduced to g=1%, then the single CPU should produce about 2,800 transactions per minute (TPM), the two-way should pull about 5,600 TPM, and the four-way (as expected) should produce 1,058 TPM (see the curve with the black dots in Figure 7).

We can also do some extrapolations and estimate the TPC-C throughput for an eight-way platform. With the g-factor maintained at 1% (or better), the expected TPC-C throughput should be about 21,000 TPM (black dots in Figure 7). How accurate is this prediction? We'll have to wait and see.

Acknowledgments

I am grateful to Doug Smucker (HP) for a discussion about the P6 bus architecture.

Notes

[1] N.J. Gunther, "NT to the Max...(NoT)," *;login:*, November 1997, pp. 9–11.

[2] N.J. Gunther, "The ABC's of TPC's," *;login:*, February 1998, pp. 50–54.

[3] J. Gray, <http://www.usenix.org/publications/library/proceedings/usenix-nt97/presentations/gray/index.htm>.

[4] N.J. Gunther, *The Practical Performance Analyst*, (New York: McGraw-Hill, 1998).

[5] N.J. Gunther and A. Cockcroft, Practical Performance Methods, Stanford class, August 1998. <http://members.aol.com/CoDynamo/Training.wics.htm>.

Source Code

```
/* gfit.c Created by NJG, 11:46:45 03-27-96 */
#include <stdio.h>
#include <math.h>
#define MAXCPU                  8
#define STARTMIN                9999.0
#define STARTERR                100.0
#define MAXITER                 10000
#define MAXSEARCH               100
double data[MAXCPU+1];
float   mpf, gf, sf;
main() {
    int         cpu, iter;
    int         findcpu1;
    double quad();
    double prev, abserr;
    double data1;
    /* input measured throughput data */
    data[1] = 2502;
    data[2] = 4039;
    printf("Measured CPU Samples\n");
    printf("CPU\tData\n");
    for (cpu = 1; cpu <= LASTCPU; cpu++) {
     if(data[cpu] > 0.0)
      printf("%4d\t%03.2lf\n", cpu, data[cpu]);
    }
    printf("\nOptimizing ");
    fflush(NULL);
    if (data[1] != 0.0) { /* no need to iterate data[1] value */
     data1 = data[1];
     findcpu1 = FALSE;
    } else { /* get the first non-zero sample to set data[1] */
     findcpu1 = TRUE;
     for (cpu = 1; cpu <= MAXCPU; cpu++) {
      if(data[cpu] > 0.0) {
       data1 = data[cpu]/cpu;
       break;
      }
     }
     data[1] = data1;
    }
    prev = STARTERR;
    for (iter = 0; iter < MAXSEARCH; iter++) {
     abserr = quad();
     if (!findcpu1)
      break;
     if (abserr < prev) {
      prev = abserr;
      data[1] += 0.1;
     }
     else
      break;
    printf(".");
    }
```

```
  printf(" Done.\n");
  printf("Best  g-factor: %3.4f\n", gf);
  printf("Estimated  X(1): %3.2lf\n", data[1]);
  printf("Cumulative error: %3.2lf %%\n", abserr);
} /* main */

double
quad() {
  int it;
  int cpu;
  double q,
  result;
  double err, min;
  float param;

  min = STARTERR;
  param = 0.0;
  for (it = 0; it < MAXITER; it++) {
   err = 0.0;
   param += 0.0001;

   for (cpu = 1; cpu <= LASTCPU; cpu++) {
    if (data[cpu] > 0.0) {
     q = data[1] * cpu * (1 - param * (cpu - 1));
     err += fabs(q - data[cpu]) * 100.0 / data[cpu];
    }
   }
   if (err <= min) {
    min = err;
    gf = param;
   }
  }

  return(min);
} /* quad */
```

the webmaster

Graft a Smart Error Page System to your Web Site

by Dave Taylor

Dave Taylor has been hacking on the Net since 1980 and has created thousands of Web pages, most of which format correctly. He's also the author of *Creating Cool Web Pages with HTML* and *Teach Yourself Unix In a Week*.

<taylor@intuitive.com>

I usually talk about standalone CGI programs in this column. But I just set up a new Web server (RedHat Linux 5.0 on a 300Mhz Pentium II box, if you're curious) and I decided that, instead of the ugly generic error messages given to people when they encounter an error on my Web site, I'd like to offer something more useful. My error message would not just have my company logo (which is a great first step, of course), but actually help people find what they seek on the site itself.

The process of adding this error page to the Web server, writing the page, and then writing a simple underlying search engine (using `grep`) is what I talk about in this column.

Hooking in Your Own Error Page

The first step is to delve into the Apache Web server configuration file. (If you're running a Web server other than Apache – which I think is fabulous – then you'll probably have to do something slightly different in this spot.)

The file, usually named `/etc/httpd/conf/httpd.conf` contains quite a few lines of different configuration elements, the vast majority of which you should definitely not touch until you're an Apache configuration expert. Fortunately, what we want to do is straightforward.

Apache Web servers can serve up lots of different domains on the same IP address and Web server. Indeed, my system is host for about 15 different Web sites. The error page I'm adding here is only for the `.intuitive.com` domain, so the trick is to find the `.intuitive.com` virtual host configuration section in the file and then add a specific line.

Before my changes, the file looked like:

```
<VirtualHost www.intuitive.com>
ServerAdmin webmaster@intuitive.com
DocumentRoot /web/intuitive
ServerName www.intuitive.com
ErrorLog /log/intuitive/error_log
TransferLog /log/intuitive/access_log
AgentLog /log/intuitive/agent_log
RefererLog /log/intuitive/referer_log
</VirtualHost>
```

This defined the actual filesystem location of the root directory of this domain (`/web/intuitive/`) and the location of all the log files (`/log/intuitive/`). To hook in the new error page, I simply added:

```
ErrorDocument 404 /error-page.html
```

somewhere in this configuration section. Where you place it doesn't matter.

Creating the Error Page

Though the previous configuration appears to have the error page in the topmost directory of the filesystem, Apache is smart enough to know already that you've specified a root in the system for the specified domain, so in fact this file needs to be located at DocumentRoot/ErrorDocument or, a bit more clearly, `/web/intuitive/error-page.html`.

Part of the goal of the error page is to offer visitors the ability to enter a keyword or two and search through all documents on the site to try and find that which they were originally seeking. That's reflected in the middle of the simple HTML document created as the error page:

```
<HTML>
<HEAD>
<TITLE>Intuitive Systems: Start Making Sense</TITLE>
</HEAD>
<BODY BGCOLOR=#000099 TEXT=#FFFFFF LINK=#ffffff VLINK=#ffffff
      ALINK=#ff0000>
<CENTER>
<IMG SRC=/Graphics/banner.gif ALT="INTUITIVE SYSTEMS" WIDTH=485
      HEIGHT=62>
<P>
<TABLE BORDER=3 CELLSPACING=15 CELLPADDING=10 WIDTH=75%>
<TR><TD ALIGN=center>
<h1>Error!</h1>
<h2>You've requested a page that can't be found!</h2>
</TD></TR>
</TABLE>
<BR><BR><BR><BR>
<font size=+3><B>Search for a specific page</B></font><br>
<HR width=75% size=1>
<FORM ACTION=/apps/search-everything.cgi METHOD=get>
<font size=+2>Enter a few key words:</font>
<INPUT TYPE=text NAME=p SIZE=25>
<INPUT TYPE=submit VALUE="Find Page">
</FORM>
<BR><BR><BR><BR>
<a href=http://www.intuitive.com/><font size=+1>[The Intuitive
Systems Site Home]</font></a>
</CENTER>
</BODY>
</HTML>
```

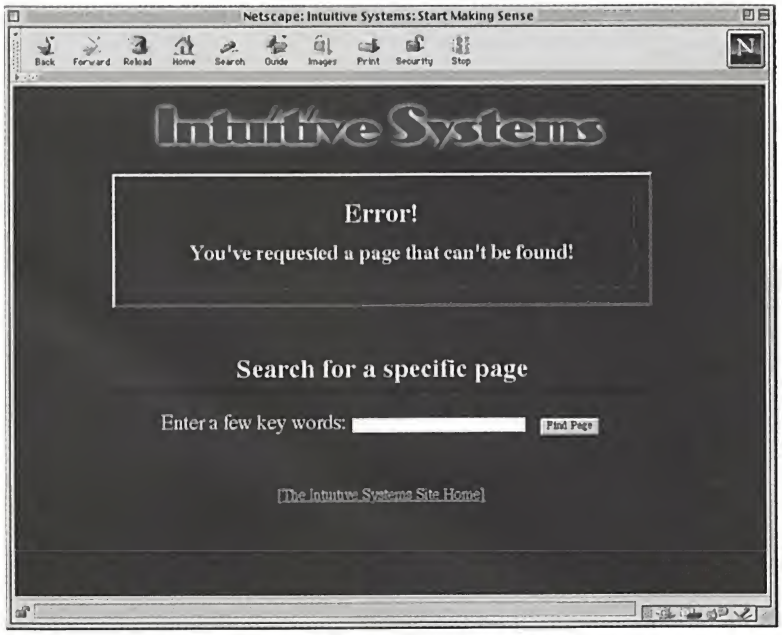


Figure 1: The new improved Intuitive.com error page

I won't belabor the HTML here – it's all pretty straightforward – other than to note that the error page includes a form that prompts for a few key words and then feeds the user entry to the CGI script `/apps/search-everything.cgi` on the server. One trick worth mentioning: explicit tables can be a nice way to box an important message on the page!

Figure 1 shows you how this page looks. You can, of course, search for some gobbledygook URL on my site and find it for yourself quite easily!

Without the search capability, we'd be done. New error page, much cooler than the default "404 File not found."

However . . .

Building a Search Engine

The good news about building a search system for your Web site is that you've already made the smart move: you're running a UNIX-based operating system. This means that you can let the `grep` command do all the work. Because we're

using a METHOD=GET in the form itself, the pattern entered is held in the environment variable QUERY_STRING. Sent as name=value, a quick invocation to sed strips it to its basics:

```
pattern="`echo $QUERY_STRING | sed  's/p=//g'`"
```

Armed with the search pattern, you then use the find command to look through all the HTML files on the site:

```
find /web/intuitive -name '*tml' -print | xargs grep -il '$pattern'
```

This gives us the ability to display matching files, and we can easily add clickable links to them all by first stripping out the actual file root (because remember that /web/intuitive/index.html is the URL /index.html) with two lines buried in a for loop that steps through the output of the above find command:

```
for filename in `cat $outputfile' ; do
  newfilename="`echo $filename | sed 's/\/web\/intuitive//g'`"
  echo "<a href=$newfilename>newfilename/<a>"
done
```

But we can do better than this and have output that's considerably more attractive and interesting. The solution is to extract the TITLE of each document by again using grep, stripping the HTML tags therein, then using that as the text of the link:

```
for filename in `cat $outputfile` ; do
  newfilename="`echo $filename | sed 's/\/web\/intuitive//g'`"
  title="`grep '<TITLE>' $filename | \
     sed 's/<TITLE>//g;s/<\/TITLE>//g'`"
echo "<a href=$newfilename>$title</a>"
done
```

There's one problem with this. Going through my Web site for a quick analysis reveals that several documents have remarkably similar titles (some of which aren't even useful, if you can believe it!). As a result, the output really needs to list both the filename and the TITLE of the document, as available.

Now all that's left is to do some error checking (What if they skipped entering a pattern? What if there are no matches to the pattern?) and wrap it in some nice HTML formatting. Again, I opt for a TABLE to have it look nice on the screen, as you can see in Figure 2.

The final CGI script, written as a Bourne shell script, is shown here:

```
#!/bin/sh -f
tempout=/tmp/searchout.$$
pattern="`echo $QUERY_STRING | \
   sed 's/p=//g'`"
#
echo "Content-type: text/html"
echo ""
echo "<HTML><TITLE>Intuitive Systems
Search Results for
$pattern</TITLE>"
echo "<BODY BGCOLOR=#000099 TEXT=white
LINK=white VLINK=white
ALINK=red>"
echo "<CENTER>"
echo "<P>"
echo "<IMG SRC=/Graphics/banner.gif
```

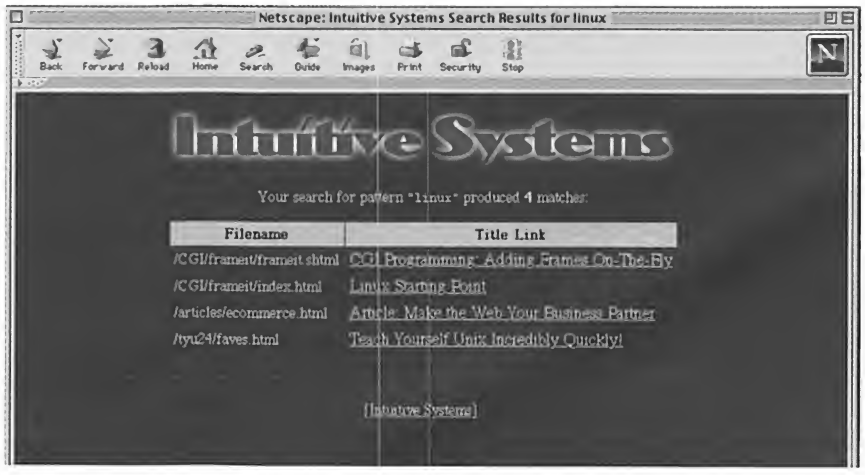


Figure 2: The results of a search for "Linux"

```
ALT=\"INTUITIVE SYSTEMS\" WIDTH=485 HEIGHT=62"
echo "    LOWSRC=/Graphics/banner-lowres.gif>"
echo "<P>"
# — no pattern entered?
if [ X$pattern = X ] ; then
  echo "No matches: no pattern entered"
  echo "<P>"
  echo "<HR width=75%><P>"
  echo "<a href=http://www.intuitive.com/>[Intuitive Systems]</a>"
  echo "</BODY></HTML>"
  exit 0
fi
#
find /web/intuitive -type f -name "*tml" -print | \
    xargs grep -il "$pattern" | sort > $tempout
#
matches="`wc -l < $tempout`"
# is there a pattern?
echo "Your search for pattern <tt>"$pattern"</tt> produced"
if [ $matches -eq 0 ] ; then
  echo "<b>zero</b> matches. Sorry.<P>"
else
  echo "<b>$matches</b> matches:<p>"
  echo "<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=4>"
  echo "<TR BGCOLOR=#CCCCCC><TH><font color=black>Filename</TH>"
  echo "<TH><font color=black>Title Link</TH></TR>"
  # now let's step through the pages, extracting TITLEs, and
  # displaying them all on-screen
  for filename in `cat $tempout` ; do
   title="`grep -i '<title>' $filename | grep -i '</title>' |\
            sed 's/<TITLE>//g;s/<\/TITLE>//g'`"
   if [ X$title = X ] ; then  # no title in document
    title=$filename
   fi
   newfilename="`echo $filename | sed 's/\/web\/intuitive//g'`"
   echo "<TR><TD>$newfilename</TD>"
   echo "<TD><a href=$newfilename><font size=+1>"
            "$title</a></TD></TR>"
  done
  echo "</TABLE>"
fi
echo "<P>"
echo "<HR width=75%><P>"
echo "<a href=http://www.intuitive.com/>[Intuitive Systems]</a>"
echo "</BODY></HTML>"
/bin/rm -f $tempout
exit 0
```

Conclusions

I encourage you to jump onto my Web site and enter a URL that you are sure won't work correctly. Try <http://www.intuitive.com/missing-page.html>. Once you're there, type in a word or two as a search pattern to see what kind of results you get.

It'd be nice to refine this further so that you could have an HTML tag in specific pages that prevent them showing up as matches to a sitewide search, and for the search results to be smart enough to show you a `META DESCRIPTION` value if one is present in the file as further information.

standards reports

by Nicholas M. Stoughton

USENIX Standards Liaison

<nick@usenix.org>

Happy Birthday, POSIX!

The international working group responsible for POSIX standards celebrated its tenth anniversary in April. The ISO/IEC JTC1/SC22/WG15 first met in April 1988 and has since then guided several of the IEEE standards developed by the Technical Committee on Operating Systems (TCOS) and then the Portable Application Standards Committee (PASC) through the process of becoming formal, full international standards.

This work has been a vital part of the whole standards effort. There is no doubt that had the POSIX standards not had this international standing, they would not have been so effective, and the term "Open Systems" would not have the same weight as it does today. And it was in 1988 that the first version of POSIX.1, describing system interfaces, was completed.

Standards for operating system interfaces are primarily developed in three places: The Open Group (X/Open, who produce the Single UNIX Specification); IEEE PASC, who produce the POSIX standards, and WG15, who take specifications from both these sources and make them ISO standards. Many have commented on the waste of effort in having three sources, and efforts are afoot to try to improve coordination between them. But of the three, WG15 is definitely regarded as the least "sexy." It does little technical development of its own (though it has been argued that it could and should do more). It has a primarily political focus to ensure that the work developed in a US body is appropriate for international consumption.

The founder and convener of WG15 since its inception has been Jim Isaak of Digital (or should I say Compaq now?). He has decided that it is time for him to relinquish this task and pass it on to a successor. But who is that to be? Jim announced his forthcoming resignation last year, but after one year of searching nobody has come forward. The job is not simple, carrying the full responsibility for seeing each of the projects through, ensuring that each step of the process is followed. It means attending two WG15 and one SC22 meeting per year, on top of any work with either of the other two bodies (primarily PASC). It is a volunteer post, needing the full support of the individual's employer. At the same time, it is a position of considerable power and influence. The convener sets the agenda at the international level and can effectively steer the future of Open Systems in this area.

So what happens if nobody volunteers? Simply put, all the current work (including maintenance of completed work) would revert from the ISO working group (WG15) to the subcommittee (SC22). WG15 itself would cease to exist. The subcommittee has no interest in holding this work directly itself. In its view, if nobody is interested in volunteering to keep the working group going, then the work should stop, and those standards completed should be withdrawn. This is, therefore, another crisis threatening the very existence of POSIX.

Jim Isaak has done a terrific job over these past ten years, and we all owe him a vote of thanks. We cannot and must not let his work go to waste. If you, or someone you know, think that you would like to know more about this opportunity, please contact me! (A detailed job description can be found at <ftp://dkuug.dk/JTC1/SC22/WG15/1082>).

Our standards Report Editor, **Nick Stoughton**, welcomes dialogue between this column and you, the readers. Please send any comments you might have to:

<nick@usenix.org>

the bookworm

Books reviewed in this column:

Chris Lewis
Cisco TCP/IP Routing
New York: McGraw-Hill, 1998.
ISBN 0-07-041088-7. Pp. 402.

Scott M. Ballew
Managing IP Networks with Cisco Routers
Sebastopol, CA: O'Reilly & Associates, 1997.
ISBN 1-56592-320-0. Pp. 334.

Berry Kercheval
TCP/IP over ATM
Upper Saddle River, NJ: Prentice Hall, 1998.
ISBN 0-13-768599-8. Pp. 202.

John T. Moy
OSPF: Anatomy of an Internet Routing Protocol
Reading, MA: Addison-Wesley, 1998.
ISBN 0-201-63472-4. Pp. 339.

Uyless Black
The Intelligent Network
Upper Saddle River, NJ: Prentice Hall, 1998.
ISBN 0-13-793019-4. Pp. 208.

Louis Rosenfeld & Peter Morville
Information Architecture for the World Wide Web
Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-282-4. Pp. 202.

Paul Ferguson & Geoff Huston
Quality of Service
New York: Wiley, 1998.
ISBN 0-471-24358-2. Pp. 266.

Patrick Chan
The Java Developers Almanac
Reading, MA: Addison Wesley, 1998.
ISBN 0-201-37967-8. Pp. 976.

Matthias Felleisen & Daniel P. Friedman
The Little MLer
Cambridge, MA: MIT Press, 1998.
ISBN 0-262-56114-X. Pp. 181.

(Continued on page 63)

by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

Two years ago, it finally became possible to legitimately possess a copy of John Lions' masterful *Commentary* on V6 (Peer-to-Peer Communications, ISBN 1-57398-013-7). Thanks to an enormous amount of work by Warren Toomey in Oz <wkt@cs.adfa.oz.au>, you can now get a V6 license. See Warren's article on page 27 of this issue.

As one of those who signed the letter to SCO, I'd like to express public thanks to Warren and the rest of the preservers. Get your V6 license (or 32V) today!

Routing

Back in February, I referred to routing as an underrepresented area of the Internet. Since then, I've received two more books on routers. They are both good, but so far no one has created a "must have" item. The two volumes at hand make an interesting contrast: the Lewis is better written (in a British style), but lacks some of the detail because it tries to satisfy the novice as well as the sophisticate. Ballew's work is up to the standard we've come to expect from O'Reilly. It is a solid, thorough piece of work that comes to grips with the day-to-day needs of setting up and maintaining an IP network. It is certainly no work for the novice.

Protocols and Networks

There are five worthwhile books this month. Berry Kercheval's "no-nonsense" guide to internetworking is just that. I enjoyed his introductory chapter. His "obligatory OSI reference model" and his capsule history of the ARPANET/Internet are correct. His style is good and he writes with descriptive strength. The last 40 pages (Research Directions, Standards Documents, Virtual Channel Connections, ATM Software, Glossary, Bibliography, and Index) are truly outstanding.

The OSPF spec was RFC1131 (1989); RFC2178 (1997) updated it. If you need to understand the need for routing and multiple paths between hosts, Moy's little book is a godsend. This isn't an easy book; it's (as the subtitle says) an "anatomy" of a protocol. Comer (in vol. 1) devotes about half a dozen pages to OSPF. Moy now supplies a full analysis and explanation.

Black's volume is an interesting one. I have a prejudice against intelligent networks, largely because the IN stems from an ITU-T recommendation (Q.1201) and because the INCM (IN conceptual , model) is founded on the OSI model and uses OSI "application service elements." Black does a fine job of explaining the models – both the ITU-T and Bellcore versions. But this isn't a book on implementing anything.

Information Architecture, on the other hand, isn't on graphics. It's about dealing with the increasing complexity of Web sites and the sorts of sites that enable virtually painless growth and emphasize navigation and ease of use. A good and painless book for those building/maintaining/using Web sites.

Ferguson and Huston have written the first book on quality of service. Their chapter on QoS and ATM (pp. 95–120) should be read in conjunction with Kercheval's book (which barely mentions QoS).

Finally, I'd like to compliment the authors and publishers of these books: they are of a size that is handleable, which means they can be read, not merely consulted.

The Beans Grind On

I keep getting more books on Java. I now have over 150 and am forging toward 200. What a cottage industry!

But I want to put in some paragraphs about only two of them. Chan's *Almanac* is a truly splendid volume. At least part of it is boiled down from Chan's two volumes on Java class libraries (with Lee and Kramer), to the point that Chan claims it to be "no bigger than a pocket dictionary." He's right. At under 1K pages, it's smaller than Heinemann's Australian dictionary, but nearly double the size of the *Concise Oxford Dictionary of English Etymology*. It's a function of the size of your pocket.

The *Almanac* is divided into four parts: Packages (describes each Java package), Classes (over 600 pages of class tables), Topics (quick reference tables), and Cross-Reference. Well done, well produced, and useful. The cover shows a tree bearing applets.

Felleisen and Friedman produced their *Little Lisper* in 1989. In 1996, we had their *Little Schemer* and *Seasoned Schemer*. Now, two more small, handy volumes have arrived. *The Little MLer* is a brilliant introduction to ML, using a relatively small subset of SML. As a language lover, I can happily recommend this. Unfortunately, their Java volume, which begins with the same words as their *Little MLer* (but diverges halfway through page 3), lacks something. I'm not quite certain exactly what it is, but I found the Java book less "satisfying" than its fellows. But the student will certainly emerge with an understanding of classes, interfaces, design patterns, and objects. *MLer* is a clear winner; *Java* less so.

Apology

In my column for February, I stated that WebSite Professional lacked "remote administration capability." This isn't true: WebSite Pro has remote administration capability, it's just not browser-based. Sorry.

Thumbs Down

I admit that I don't like Microsoft's AFC (Application Foundation Classes). This might be because I don't like Microsoft in general, but I think it's because AFC is such a clear ripoff of the JFC. (They were announced on the same day; but Microsoft didn't get around to releasing AFC until some months after JFC. Hmmm.) AFC is quite complex, but JCF is complicated, too. AFC is also ugly. I guess this book and CD-ROM by Swildens and Sol might be useful if you were trapped in a Microsoft environment without a compass or a rifle. You know, "If it's that complicated, it must be wrong."

In my April 1995 column, I said that I thought Lyn Dupre's tome was cutesy, without having the ability to actually improve anyone's writing. I read about a quarter of the new, unfortunately unimproved, edition. (I find it bizarre that Dupre recommends punctuation at odds with most other North American style sheets. If you really want something on writing, I'd suggest Strunk and White, *Elements of Style* [1959 and still in print]).

More books reviewed:

Matthias Felleisen & Daniel P. Friedman

A Little Java, a Few Patterns

Cambridge, MA: MIT Press, 1998.
ISBN 0-262-56115-8. Pp. 180.

Eric Swildens & Selena Sol

Programming with Microsoft AFC

New York: Wiley, 1998.
ISBN 0-471-24891-6. Pp. 573 + CD-ROM.

Lyn Dupre

BUGS in Writing, rev. ed.

Reading, MA: Addison-Wesley, 1998.
ISBN 0-201-37921-X. Pp. 666.

book reviews

Henry Spencer and Dave Lawrence

Managing Usenet

O'Reilly & Associates, 1998. ISBN: 1-56592-198-4. Pp. 492. $32.95

Reviewed by Nick Christenson
<npc@jetcafe.org>

For years, there has been quite a pronounced gap in the literature on the operation of USENET news services. It's hard to say for sure why this has been the case. Perhaps none of the people qualified to write such a book has had the time to do so. Perhaps the list of people who actually understand the issues thoroughly enough to be comfortable tackling such a project is much shorter than might be expected. Perhaps everyone was just waiting for someone else to do it. Of course, it might be a combination of these factors, but at least there finally are books available to help the aspiring netnews administrator. Of these, the one most eagerly anticipated, a title it has held for longer than the authors probably care to think about, is *Managing Usenet* by Henry Spencer and Dave Lawrence.

The book begins with an introduction that is complete, although it might go by a little quickly for those who have no or almost no experience with USENET. A brief background on USENET is given, terms are defined, and a basic introduction of the requirements for running a news server are presented. The authors are well aware that this data will be immediately out of date. Fortunately, they do a reasonable job of impressing this upon the reader.

The second chapter covers the preparation for running a news service. This is very useful, because there is much to consider before one installs the software and starts up a feed. The third chapter is on basic operations, those that are not specific to a particular software package. For example, one must consider which control messages will be honored, what the expire policy will be, what hierarchies and distributions will be accepted, etc.

Chapters 4 through 7 cover the acquisition, installation, and basic operation of the C news distribution. This information is detailed enough, but I felt more nostalgic than informed by it. Although the C news distribution continues to be maintained (by one of the authors of this book) and there are still sites that run it, I can't in good conscience recommend to anyone aspiring to set up a new news server and who has no experience with any package to install C News. I can believe that in some environments it's not worth the effort to switch to INN from C News, but I really can't imagine someone being better off not running INN or one of its variants.

Chapters 8 through 10 cover the same ground for INN. The information here is fairly up to date, although INN development is so active right now that published information is immediately obsolete. These chapters are the best I've seen on the issues involved in installing this package. Chapter 10 covers some guidelines on which news reading clients to install and support. This is good advice, but some important clients, Agent, for example, aren't covered at all. In my opinion, way too much consideration is given to reading news off of a filesystem on the news server itself rather than via NNTP, but that's just me.

Chapter 11 covers network management and related issues. Especially interesting is the section on legal issues, at least if the news server under consideration is within the United States, but a number of folks I know with much specific expertise in this area have stated different views than the authors present here. As the authors note, very little of this debate is based on established case law, so I would have liked to see a little more consideration of other viewpoints.

In chapters 12 and 13, details specific to running leaf and hub nodes are considered. Chapter 14 covers gatewaying between USENET and electronic mail. Chapter 15 discusses the details of moderating newsgroups. Chapter 16 covers some thoughts on USENET namespace.

The next chapter provides a brief history of USENET. Several accounts of this have been written, but none gives the reader a better glimpse into what actually took place. I especially like the fact that multiple sides of some of USENET's oldest political battles and flame wars are presented in what I consider a very balanced manner, while maintaining the mood of the folks involved in these decision-making processes and of USENET as a whole at the time.

Chapter 18 dissects the anatomy of a news article, which is a very worthwhile commentary because the RFCs and the implementations have diverged so much. The book ends with what I think is one of the weaker chapters, on the flow of news traffic.

Given the pedigrees of its authors I had hoped this book would be an instant classic, that it would be the last word, at least until the protocols are changed and the software is updated, on the topic of USENET news. Although this is certainly far and away the best book on this topic, it's not quite as strong as I had hoped. Certainly, though, every journeyman news admin will benefit greatly from reading this book, and almost all veteran USENET administrators will want to have it on their shelves.

I have to admit that I was disappointed by several things about the book. First, almost no information is given on tuning USENET servers. A reasonably busy news server has a very different sort of load placed on it than other types of Internet servers. Although this is acknowledged, even the most widely repeated recommendations on how to maximize a news system's throughput are not mentioned.

Additionally, many of the sections repeat information found elsewhere, and parts of the book don't flow very well. Several times the authors present what might be regarded as a general truth, even though

there are important exceptions. These are invariably mentioned later, but I'm of the opinion that these caveats should be mentioned up front. For example, in chapter 8, they suggest one might want to renumber the active file after adding each new newsgroup. Only much later do they mention that the renumber might take several minutes if the active file is large and that innd won't respond to any requests while this is taking place. On a news server taking a significant fraction of a full feed, if newsgroup addition is not done all at once, this is a very bad idea.

Overall, much of the information in the book seems inconsistent to me, as if different chapters were written years apart or without much consideration of what was being written in other chapters. It almost seems as if it was decided that, although more editing and updating needed to be done, the book was shipped as is to the printer. But if the alternative was another year of waiting for it to come out, then shipping it now would have been the lesser of two evils.

Capsule

This isn't the ultimate word on the topic but *Managing Usenet* is the best book available on running a USENET news server, and is a worthwhile addition to the library of anyone with an interest in this topic. Although the book has some significant faults, the information it contains is easily valuable enough to compensate for them.

Eric Pearce

Windows NT in a Nutshell

O'Reilly & Associates, 1997. ISBN 1-56592-251-4. Pp. 348 $19.95.

Reviewed by Carolyn J. Sienkiewicz

<cjs@chokey.mo.md.us>

Windows NT in a Nutshell: A Desktop Quick Reference for System Administrators can be a life saver. If you are new to administering NT, it can be difficult to remember where to go to configure some esoteric option. *Windows NT in a Nutshell* is another in a continuing line of fine O'Reilly quick reference books. It provides a concise and clear roadmap to the NT system.

Eric Pearce has written this book in a simple and straightforward style. There is a great appendix that takes the mystery out of which service, protocol, and administrative tool components are workstation and which are server. In one index, there is a table of general tasks you might want to work on, with cross-references for the section of the control panel to access it, the admin tool to configure it, and the command line version as well.

The control panel and administrative tools are covered in a detailed reference format. For each top-level selection (or tool), the author gives you

1. What it requires (NT server? backup device?)
2. A summary of the task or topic
3. A pull-down menu layout of subtasks or subtopics where you can go from the the top-level selection
4. A short description of each possible selection instance of number 3 above

Between the top-down roadmap offered in the control panel and administrative tools sections and the task index I already mentioned, you should be able to find just about anything.

My favorite section of the book is "Using the Command Line." For one thing, I like it because I like to use the command line. But aside from that, I particularly like this section because the commands are presented in man page style. The author follows the traditional format of:

usage
command definition
options
examples
see also

This approach has worked beautifully for UNIX types for years because of its simplicity and thoroughness, and it is helpful here because it is reassuring to see the NT information presented in a familiar format. It makes it less a foreign universe and more "just some more man pages."

In a departure from pure reference, there is also a substantial section of hints, tips, and general troubleshooting advice. This, in combination with the highly effective reference job Eric Pearce has done, makes this book a highly recommended resource for every NT administrator (or *potential* administrator). And when you do eventually know NT so well that you no longer need it, the next new admin will be very glad to get the use of your dog-eared copy.

Brian L. Wong

Configuration and Capacity Planning for Solaris Servers

Prentice Hall, 1997. ISBN: 0-13-349952-9. Pp. 428. $45.00.

Reviewed by Nick Christenson
<npc@jetcafe.org>

There are several good books available for system administrators who want to know how to solve specific problems they may encounter, but there had been a notable lack of information on how to specify and build systems to begin with. As the title suggests, *Configuration and Capacity Planning for Solaris Servers* attempts to do just this, providing information on how to determine what gear to obtain and how it should be set it up to achieve certain ends.

The book is divided into nine chapters, the first of which is titled "Methodology" and defines terms and concepts that will be used later in this book. It's a pretty decent introduction, as it is absolutely necessary that the reader and author agree on the terminology to be used.

The next four chapters discuss configuring different kinds of servers using Sun gear, although this information is immediately applicable to at least some other UNIX systems. Advice on designing and benchmarking NFS fileservers, database servers, various flavors of Internet servers, and timeshare systems are all discussed here. Wong advises the reader on how to estimate various kinds of load and then on what (Sun) gear to buy in order to achieve these ends. This is remarkable. As far as I know, this is the first time such a comprehensive guide to determining resource needs has appeared in print. Wong's advice on how to benchmark these systems and how to design a system to meet these needs is generally good, or at least I agree with most of it. For example, the author and I share opinions on the importance of small disks as opposed to large disks for filesystem throughput, the common overemphasis on CPU speed, and other system design principles that have rarely been published.

However, once the author begins discussing specific cases and recommendations, I start to have disagreements with his conclusions. Just one example is on the configuration of USENET news servers, a topic I happen to know a bit about. One simply can't make good recommendations about what sort of gear one needs without knowing how long expire times are going to be, at which speeds clients connect, what a typical client's reading profile looks like, etc. Making recommendations without this information is, in my opinion, simply foolish; and Wong does this in several areas, often using logic with which I cannot agree. Despite this, if one is incapable, for whatever reason, of calculating these metrics oneself, then I'm willing to concede that Wong's numbers would be considerably better than nothing; but I urge folks to take these numbers with a healthy dose of salt. In any case, I'm fairly sure the author would agree that one is always much better off doing all the requirements gathering oneself rather than depending on anyone's numbers.

This leads to another problem I have with the book. Although I expect a book written by a Sun employee and published by Sun Microsystems Press to be pro-Sun, in my opinion this occasionally gets out of control. For example, how many folks need advice on how to use a Sun box as a high-speed router? Sure, sometimes they come in handy as low-speed gateways, and I admit that I know a couple of places where they do mess with gated; but if your Sun runs out of capacity as a gateway, the best advice for most sites is, "buy a bigger dedicated router." Similarly, even at the time this book was written, if "buy a SPARCserver 1000" is the correct answer, I can't imagine what the question could possibly be. The author seems to be reaching for a reason for someone to buy each piece of gear in Sun's catalog at the time. In one case, the application requires low-performance, but several SBus, slots. Sure, one could do this with a SPARCserver 1000, but it would be cheaper to do it with multiple Sparc 5s or 20s.

Another related complaint is that when the book was written, Sun was overhauling its product line with the release of the UltraSparc chips. A lot of the recommendations in this book were wildly out of date at the time of its publication. Also, no thought was given to weighting recommendations by architecture lifetime, and this, in my opinion, undermines the author's credibility in making other recommendations. For example, even if I didn't need the horsepower now, I'd have to recommend the purchase of an Ultra 1 over a Sparc 10, despite the slightly higher cost, because the product has a longer life. Thus, perhaps surprisingly, this book may be more valuable to folks who work in non-Sun shops, who would not be tempted to take this advice on its face.

The last four chapters are on Sun system architecture, storage system architecture, backup systems, and Solaris 2. These chapters are all extremely good. This book contains the best single, simple source of information explaining the various available RAID systems, including their relative strengths and weaknesses (although I have some minor quibbles with a little of the information) and tape backup systems I've seen. In fact, these chapters make the book worthwhile by itself.

It sounds like I'm being very hard on this book, and I am, but it's not because I think that the book shouldn't be read. On the contrary, I think it's one of the more important books for system administrators written in the last few years. It is because this book is so important that I'm being so critical. This is the first book of its type, and it is well worth reading, I

just feel that it should be read critically. The topics of the book intersect sharply with my areas of expertise and areas of strongest opinion; others may not have as harsh a reaction as I had.

Capsule

Despite some weak points, a stronger bias toward weaker Sun solutions than is warranted, and some questionable advice on some specific implementation recommendations, this is a very important book that covers much ground for the first time anywhere. The sections explaining how systems work and how to measure requirements are extremely good, but the specific recommendations are both out of date and should be viewed skeptically in any case. Nonetheless, I recommend the critical reading of this book by everyone involved in the design and specification of computing systems.

Nick Heinle

Designing with JavaScript: Creating Dynamic Web Pages

O'Reilly & Associates, 1997. ISBN 1-56592-300-6. Pp. 241. $29.95. CD included.

Reviewed by Bruce O'Neel

<beoneel@acm.org>

One sign that you are getting older is that the book authors start to seem quite young. The other sign is that you can't imagine that someone that young could write a book, yet alone a reasonable one. Well, Nick Heinle at 17 is less than half my age, but the second prejudice isn't true. This book is good for its target group.

Rather than at programmers, *Designing with JavaScript* is aimed at Web designers. What is included is a lot of code examples to help you add JavaScript to your Web pages. What is not there is an overview or detailed description of JavaScript. If you are a language person, say, one who reads the *C++ Annotated Reference Manual* for fun, this is not a good book for you. If you'd like to add JavaScript to Web pages without having to understand more than is necessary, then this is a good book. Given that this is a fluid field, it's tough for a book such as this to remain current, so you'll also find yourself referencing the author's Web site at <http://www.webcoder.com>.

Overall, the explanations are clear, and Mr. Heinle understands that although he's talking to a group of people who possibly don't know about programming, they are not idiots. This is a book that teaches by example, and there are many examples, all of them highlighted in what in the X world I believe would be sea foam green. The book is much more understandable if you already have a good grasp of HTML because, without a good knowledge of HTML, most of the examples will not make a lot of sense. Additionally, you would have a hard time understanding the motivation for the different examples.

Although the examples are clear they suffer from what could be called FORTRAN 66 formatting. A little indentation wouldn't have hurt things for those of us who are used to it. Plus it wouldn't have hurt to show a good formatting style to a group of people whose first experience with programming will be this book. *Designing with Java Script* also has many highlighted regions that explain more about JavaScript; and, unlike many books, the highlighted sections are quite important to understand JavaScript, and therefore the book itself.

Chapter 1, "Diving into JavaScript," starts right off with some neat JavaScript additions to your Web pages. Each chapter includes a little "In this Chapter" list that lets you quickly decide if this chapter is going to be useful to you. Chapter 1 covers some very basic JavaScript commands such as writing to the currently displayed page, writing to the message window at the bottom of the browser, and some overviews of how the JavaScript object model works. It also includes how to hide JavaScript from older browsers. This is probably the toughest chapter to get through if you aren't a programmer already because lots of programming concepts are introduced in short order. If you can get through this chapter, then the rest of the book shouldn't be too bad.

Chapter 2 covers how to control browser windows, write functions, and create remote controls. A remote control in this case is a detached browser window with buttons that help you navigate around the Web site. Chapter 2 is the first place where the book talks about the differences between various Netscape Navigator (NN) versions, just one of the problems that adds complexity to one's life as a JavaScript programmer.

Chapter 3 tells you how to make JavaScript and frames work together. Yes, those dreaded frames can be made to work better if you are willing to do some programming. Here you also learn how to make expanding and collapsing titles and subtitles in your navigation box.

Chapter 4 shows you how to make JavaScript and forms work together. This is one place where JavaScript can really help because you can have JavaScript do the validations on the client, rather than connecting back to the host for the validations. There also is a long discussion of functions in JavaScript, with a number of good examples to show you practical uses of functions.

Chapter 5 covers arrays and the interesting things you can do with them in JavaScript.

Chapter 6 talks about all the hoops you'll have to go through to make sure you do the right thing, given different browsers. One of the things that makes JavaScript a bit difficult to program in is that each version of NN and Microsoft Internet Explorer (MSIE) implement a slightly different version of JavaScript with different bugs. In fact, in my tests with NN

4.04 on Solaris, I found that a number of the examples didn't quite work as expected. I hope this will settle itself out now that the European Computer Manufacturers Association has started standardizing ECMAscript.

Chapter 7 covers dynamic images. Dynamic images are images that change as different things happen on your Web page. An example of this is when you move your cursor over some part of your Web page, an image changes from a static image, say, of a dog, to a dog wagging its tail. This chapter also talks about preloading images so that when you do move your cursor over some bit of your Web page, something happens beyond trying to connect to a Web site.

Chapter 8 will be a favorite with the privacy folks. Here's where you find out more than you ever wanted to know about cookies from the JavaScript side of things. You'll get to see how to make a welcome mat for a Web site that is displayed the first time a user arrives, as well as forms that remember where you were the last time you visited. Although cookies are controversial, using JavaScript and cookies together at least keeps all the information on the client computer, rather than keeping it on the host computer, which might help with privacy concerns.

Chapter 9 discusses Dynamic HTML (DHTML) and Style Sheets (SS), two new features added in version 4 of NN and MSIE. Here NN and MSIE diverge quite a bit, so this chapter is in two parts, one for MSIE and one for NN. This is a long and involved chapter because there is a lot to learn about DHTML and SS.

Chapter 10 talks about a concept that NN introduced in the first beta of NN called layers. This was superceded by Dynamic HTML and Style Sheets, but this chapter is included for the sake of completeness. It's probably best not to use layers because there are now better ways to accomplish the same task.

Chapter 11 is devoted to a Dynamic HTML application called The Show. The Show is a multiple-choice game, and this is a good example for Dynamic HTML. The downside is that I was not able to get this to work on NN 4.04 on Solaris.

Chapter 12 is a wrap-up chapter called "Advanced Applications." It first talks about user-defined objects. Next it shows three examples of a quiz application, a Web tour, and relational menus. Relational menus are menus where one menu changes in relation to what one choose from the other menu.

The book includes a CD-ROM with all the examples from the book plus a demo version of Acadia's Infuse, a JavaScript programming tool. This being an Internet tool, it's been sold, in this case to NetObjects, and is now called NetObjects ScriptBuilder 2.0. Because there is no Solaris version, I did not get a chance to test it.

This book will work best if you work through it while sitting at a system with an editor and a Web browser up so that you can try the examples as you read. Most of the examples aren't very long, so they aren't going to take too long to enter even though they are on the CD-ROM. If you are new to a language, actually typing the examples in can be a big help in learning that language. It's amazing what the eye misses that the constant banging of finger on key will reinforce.

So, that all important question, is it worth the $30 of your hard earned money? If you're not a programmer and want a fairly easy introduction to JavaScript programming, then yes. If you are an accomplished programmer already, you probably will be happier with a more in-depth book. If you do decide to purchase it, you should work through it as quickly as possible. Time flies in the Internet world, and if you wait too long, the book will become outdated.

David H. Freedman and Charles H. Mann

At Large: The Strange Case of the World's Biggest Internet Invasion

Simon & Schuster, 1997. ISBN: 0-684-82464-7. Pp. 315. $24.00.

Reviewed by Nick Christenson
<npc@jetcafe.org>

Since Cliff Stoll wrote *Cuckoo's Egg* in 1989, we've been inundated with similar books about wily hackers, their exploits, and how they were finally caught. *At Large* is another one of these, detailing the exploits of Phantom Dialer, a kid from Oregon who broke into many hundreds of computers over the Internet in the early 1990s.

The book is not without its share of hyperbole, subtitled *The Strange Case of the World's Biggest Internet Invasion.* The liner notes claim that the book "is the astonishing, never-before-revealed tale of perhaps the biggest and certainly the most disturbing computer attack to date." Although the events are interesting and the account is reasonably well written, the story certainly doesn't live up to this hype. Yes, the perpetrator broke into a lot of computers. Yes, the book points out the inherent vulnerabilities of the Internet, although no more definitively than any other book of the genre. Yes, some folks lost a fair bit of sleep over these incidents. But on the "disturbing" index, this story doesn't crack the top ten.

I would have reacted more positively to this book if it weren't for all the hype it presented. I feel it's implied that we're going to be blown away by the exploits related here, but let's face it: Phantom Dialer didn't do truly massive amounts of damage to computer systems all over the world, although he certainly could have. He just broke into them. It wasn't the case that nobody could track him down; it's just that (1) the legal system wasn't ready at that time for a case like this, (2) law enforcement wasn't interested

because they didn't understand the threat, and (3) many, if not most, of his victims didn't care much that they had been penetrated. Big, yes, a problem for many people, yes, the "most disturbing computer attack to date," sorry, no.

Additionally, the book fails to discuss any of the significant changes that have been made in the laws and law enforcement that make Phantom Dialer a less significant threat today than six years ago. Although Internet security is still woefully inadequate, exactly this sort of invader is much less likely to get this far or last this long. I'm not saying that the Internet isn't vulnerable or that these sorts of attacks won't work, but there are more folks paying attention to security these days, and their remedies are more rapid and precise. It is much less hard to keep a Phantom Dialer out of one's network these days than it was. There are still very significant threats to one's networks, but this guy isn't one I'm worried about.

Still, it is an account of a hacking/cracking story that heretofore had not received much, if any, public distribution. It's a story that's worth hearing, but quite honestly, the important parts of the book could have been related as an article in *Wired* and not lost anything. Add *At Large* to the list of unremarkable, although by no means embarrassing, similar works.

I would guess that reactions to this book will be mixed. Those with any understanding of the state of Internet security won't be surprised or shocked by anything in the book. Those who are interested, but nontechnical, may be shocked and appalled. If they are, so much the better for the state of the Internet. Those involved with the hacking/phreaking community will probably feel that the book is another lame attempt by the conventional press that fails to reveal anything worthwhile about what they're really like. I would have no counterarguments for any of these claims.

At its heart, this book is yet another unremarkable attempt to duplicate the greatness of *Cuckoo's Egg*. I found it mildly entertaining to see how folks I know are depicted, and it does help make some sense out of the CERT advisories circulated at the time, but that's about it. Read it if you feel you must, but don't expect greatness.

Capsule

If you enjoy reading every book of this genre, you'll find it about as good as most. If you were involved in Internet security in the early 1990s, this book will help explain some of what was going on and why. Unfortunately, the book in no way lives up to the hype on the jacket. It's another passable, if unremarkable, story of computer security violations riding the coattails of the excellent *Cuckoo's Egg*.

Syd Logan

Developing Imaging Applications with XIELib

Prentice Hall, 1998. ISBN: 0-13-442914-1. Pp. 668. $54.00.

Reviewed by Daniel Lazenby

<dlazenby@ix.netcom.com>

XIE? What is that? Maybe it has something to do with X? Why would I want to use XIE anyway? These were just a few of the questions that ran through my mind when I first read the book's title.

The core X specification's ability to display or enhance images is limited to three uncompromising formats. By design, core X ships images uncompressed across a network. It does not support imaging industry standard formats or standard image compression algorithms. Each image must be matched to a given display type. Image work has to be done at a very low level of abstraction.

Unfortunately for X, the real world of imaging includes many other file formats, encoding, and compression techniques. In response to these library limitations the XIELib API was created. XIE is a set of X image extensions (XIE). Based on X11R6, XIE was released three years ago along with X11R6. XIE capabilities include:

- tools that permit the display of any image on any X display type
- offload client processing (image decompression) to the X display
- read and write JPEG and CCITT Facsimile (FAX)
- providing X display image manipulation and enhancement primitives
- foundation for extending encoding and decoding functions to other imaging industry standard formats

Thumbing through the book reveals pages and pages of code examples, samples, and a CD-ROM that contains all of the code and examples listed in the book. Descriptions and release notes for each client are provided in CD readme files. The <http://www.users.cts.com/crash/s/slogan> Web site contains current versions of the book's code. By the way, the author took time to point out that the sample client code was written and compiled using GNU tools on a Linux-powered box.

The book is nineteen chapters long and opens with a quick review of image processing. It then moves on to image processing capabilities of Xlib APIs. The third chapter begins with an overview of XIE and its architecture, then moves into a detailed presentation of sample XIE client code. This chapter includes an introduction to photoflos. A Photoflo is a graphical method for illustrating the sequence in which XIE element operations will be done on an image. Each photoflo element uses an X resource ID. A limited number of resource IDs are available to X clients. An environment called Photospace was invented to overcome this X resource ID limitation.

In Chapter 4, the author steps through the process of photoflo creation, specifying photoflo elements, transmitting

photoflos to a server, and releasing memory and resources used by the photoflo. This chapter then presents when and where to use many of the 30+ XIELib functions. Having looked at the big picture, major chunks of the remaining chapters explain photoflo elements and the specifics relating to using the XIELib API in considerable detail. The book is heavily laced with code that supports the idea or example being presented. Chapters 16 and 17 describe specific XIE techniques for encoding and decoding JPEG Base, JPEG Lossless, CCITT G31D, CCITT G32D and 42D, TIFF2 and TIFF packets file.

Several color and black and white pictures illustrate the effects produced by the various XIELib functions. Most examples use the same picture to illustrate the results of a specific XIELib function. I feel this permits a more accurate comparison and understanding of what can be done with the XIELib functions.

According to the author, the purpose of this book is to provide better documentation of XIE, guidance on writing XIE clients, and to provide sample code. Having read this book, I feel I have a better understanding of XIE and how XIE clients are written, and the book and CD-ROM are loaded with sample client code.

Whitfield Diffie and Susan Landau

Privacy on the Line

MIT Press, 1998. ISBN: 0-262-04167-7. $25.00.

Reviewed by Rick Umali
<rgu@world.std.com>

On April 4 1998, *The New York Times* front page reported that Douglas Groat, a CIA technician, was charged with spying. This disgruntled employee, dismissed after years of service, told two foreign nations how the United States had spied on them. Groat was involved in "the penetration of cryptographic systems of foreign governments." Revealing which two nations "would gravely endanger American national security," said the Justice Department.

Whitfield Diffie and Susan Landau's book, *Privacy on the Line*, attempts to shed light on the complex and fascinating field of privacy, cryptography, wiretapping, and encryption. The book puts context around stories like Mr. Groat's. by arching over the broad history of the United States government's early formulation of privacy policy, introducing many of the technical concepts, and finally bringing these together into an argument for more cryptography.

Privacy on the Line gave me a great introduction to this field. If you're like me, you've dabbled in security (perhaps you've created a public-key "just to see what it's all about"). Maybe you have used a SecureID card at one time or another. And, if you're like me, you probably are put off by the government's Clipper program, but perhaps were not sure "what it's all about." The book provides excellent background on the technologies (encryption, wiretapping) that our government debates and attempts to set policies about. The authors also attempt to provide the government's motivation for penetrating the cryptosystems of foreign governments and for having access to keys via escrow programs like Clipper.

The book is not easy to read, primarily because it is dense with information. The bibliography itself is 31 pages. The "notes" section is 40 pages, and you will be referring to it often. One thing missing is a decent glossary. Using the 24-page index to look up terms makes for a lot of back-and-forth between pages. Part of what makes the book heavy reading is the recitation of government activities, laws, acts, and various agencies surrounding cryptography and privacy, which march across the pages for historical background. It's difficult to keep track of all the acronyms without resorting to notes; a timeline of the key acts and the creation of agencies would have been a welcome addition.

The chapter introducing the different technologies is titilating. You will read about the different secure telephone units (STU-III), code "generators," and true public key facilities (for "manufacturing and distributing" secret keys). You will read about intelligence, how it's defined, how it's gathered, and where cryptography fits in. You'll learn that the largest phone tap was 12 tons (p. 259). You'll read about how US export law prohibits the shipment of the source code to Bruce Schneier's book, *Applied Cryptography* (p. 107), and how the government can consider this source code a "weapon of war (a munition)."

The chapter on National Security and Law Enforcement provides numerous "creepy" parts, such as agencies charged with Signal Intelligence using planes and other "provocations," forcing secret radar stations to reveal themselves (p. 258), and the use of infrared imaging to record "the

veins in the face" for electronic surveillance (p. 119). I half expected Tom Clancy or some other techno-spy author to make an appearance in these chapters.

Activities by our government, and others, seem hidden behind the euphemism of national security. It was disturbing to read about the paranoia of our own government, especially during the Kennedy/Johnson/Nixon era. The authors provide many examples of governement-violated privacy.

One subject that doesn't get a lot of attention is the use of crypto systems by commercial entities. Diffie and Landau state that the commercial market for cryptography "already outstrips the military market" (p. 6), but then leave it at that. They describe how cryptography can provide "digital equivalents" for human interaction (p. 45), and they _footnote a Citicorp banking system "breach" (footnote, p. 251), but for the most part, the book stays within the realm of public policy, even though the private sector may provide the impetus for changing our government policies on cryptosystems.

Cryptography lies at the heart of our daily commerce: access to our bank accounts, to cable television, to electronic services. All of these activities rely on crypto, and its use of cryptography must remain unfettered. Commercial companies' sensitivity to their customers' privacy will be a factor when the government attempts to control the means of this privacy. Although they touch on this, the authors do not cover this territory fully enough.

Their thesis is that privacy, as defined by people talking without fear of being overheard, is hardly possible in today's world. Too many paths need to be crossed before two people can "find a safe quiet spot" to discuss matters in private. Today, these two people (your family members, your colleagues) may discuss matters over the phone, the Internet (email), or via fax, and there's no guarantee that these discussions are private. The authors contend that "we must build the means of protecting that privacy into our communication systems."

But the enemies of privacy seem to be the very public entities that we pay taxes for: our police and military. Hence we launch full bore into the debate over whether the government should have a "back door" into "private keys" (Clipper), or whether the government can compel a video store to reveal our video rental habits (p. 268, in reference to Supreme Court candidate Robert Bork), or other matters of privacy. Just how much should the US government know about our lives?

This debate quickly fosters extreme positions. The book makes no attempt to be objective. The conclusion is a plea to let cryptography grow. There are other ways for the government to insure the national security. Instead of spying on our foreign neighbors, and penetrating their crypto-systems, we can make use of other "intelligence." Diffie and Landau argue that modern wiretapping has done little to prevent or prosecute criminal activity (despite what movies and television would indicate).

Diffie and Landau plead that privacy is a fundamental human right (p. 126), and that our country is different from totalitarian countries because the United States does not violate those rights. So why not give us more "privacy" tools? What is the government afraid of? The tension in these arguments and questions is palpable. The authors end the book with a section titled "Suppose We Were to Make a Mistake?" in reference to allowing unrestricted growth of cryptography. Does the government care to let us make these mistakes? Can we all breathe easier, knowing that criminals, and "bad" governments, have access to crypto-systems that the US Government can't break?

Diffie and Landau ask us all "what kind of a society do we want to be?" They hope the answer is "a country that allows us access to privacy" and systems that maintain that privacy.

USENIX news

Election Results

The results of the election for Board of Directors of the USENIX Association for the 1998-2000 term are as follows:

President:

Andrew Hume	1,153 + 66 abstentions

Vice-President:

Greg Rose	1,133 + 87 abstentions

Treasurer:

Daniel Geer	1,133 + 78 abstentions

Secretary:

Peter Honeyman	1,103 + 110 abstentions

Directors:

Elizabeth Zwicky	1,004
Jon "maddog" Hall	768
Pat Parseghian	559
Hal Pomeranz	520

Not elected for Director:

Jay Lepreau	483
Jordan Hubbard	462
Darrell Long	374
Jim Duncan	264
Mark Teicher	158

Total number of ballots cast: 1,245
Total number of invalid ballots: 2

Newly elected directors will take office at the conclusion of the next regularly scheduled board meeting, which will be held June 14, 1998 in New Orleans, LA.

Ellie Young
Executive Director, USENIX Association

Notice of Annual Meeting

The USENIX Association's Annual Meeting with the membership and the Board of Directors will be held June 17, 1998, at the New Orleans Marriott Hotel, site of the 1998 USENIX Technical Conference. The meeting will begin at 8:00 p.m., and the room location will be published in the conference directory. This is a great opportunity to get your questions answered and make suggestions on how we might serve you better. Everyone is welcome!

USACO Spring Tournament Results

by Rob Kolstad

Rob Kolstad, editor of *;login:* and president of BSDI, is head coach of the USA Computing Olympiad Team.

<kolstad@usenix.org>

The Spring 1998 USACO Tournament included no fewer than 126 entrants competing from a total of 15 countries. Three contestants scored a perfect 1,000 points:

Grade	Age	Country	Name
11	16	Latvia	Janis Sermulins
12	18	China	Peng Li
12	18	Indonesia	Andy Kurnia

The top 15 scorers (those above 800 points) included nine seniors, four juniors, and one freshman. This was the most difficult contest that we've posted so far on the Internet. Missing just two little test cases cost over 100 points.

Almost half of the entries came from outside the USA – way up from previous contests. Here's the distribution of the top 25 scorers:

Country	# of Top Scorers	Percent
USA	9	36%
Rumania	5	20%
Latvia	3	12%
Indonesia	2	8%
Netherlands	2	8%
Poland	2	8%
China	1	4%
Greece	1	4%

Note that many countries performed far better than "random": Rumania has 4.8% of the entrants but 20% of the high scores, for example.

Here is problem 5, "STRINGSOBITS," proposed by the Dutch coach, Kim Schrijvers:

Consider an (ascending order) ordered set S of strings of N (1 <= N <= 32) bits. Bits, of course, are either 0 or 1. This set of strings is interesting because it is ordered and contains all possible strings of length N that have L (1 <= L <= N) or fewer bits that are "1."

Your task is to read a number I (1 <= I <= sizeof(S) and also I <= 2147483647) from the input and print the Ith element of the ordered set.

Input Format

The input file contains a single line with three space-separated integers, N, L, and I.

Output Format

The output file should contain a single line with a single integer that represents the Ith element of the set of integers of length N bits with no more than L bits that are "1."

Sample Input (file `INPUT.TXT`)

```
5 3 19
```

Sample Output (file `OUTPUT.TXT`)

```
10011
```

SANE'98 in November–An International Conference

By Jan Christiaan van Winkel

SANE Program Co-chair

SANE'98 (System Administration and Networking) conference will be held November 18-20 in Maastricht, the Netherlands. The conference is being organized by the Netherlands Unix Users Group (NLUUG), co-sponsored by USENIX , and Stichting NLnet.

Program

The program will include two tutorial tracks on November 18. On November 19 and 20 two keynotes and two tracks of presentations will be held on these subjects:

- Security tools and techniques
- Managing enterprise-wide email (what about Unsolicited Commercial Email – UCE?)
- Experiences with free software, including operating systems, in a professional environment
- Innovative system administration tools and techniques
- Distributed or automated system administration
- Incorporation of commercial system administration technology
- Adventures in nomadic and wireless computing
- Intranet development, support, and maintenance
- Integration of new networking technologies
- Integration of heterogeneous platforms
- Performance analysis, monitoring, and tuning
- Support strategies in use at your site
- Effective training techniques for system administration and users

International Character

Speakers of many nationalities will be participating, and participants are expected from Belgium, Germany, Denmark, the UK, and the US. NLUUG is announcing the conference through European and American UNIX user groups. The conference language will be English.

Exhibits

Vendors, including a number of book stands, will present their products.

Advance Registration

In order to relieve the registration desk at the expected rush hours in the mornings

WEB SITE

http://www.usenix.org

MEMBERSHIP

Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS

Eileen Cohen
Telephone: 510 528 8649
Email: <cohen@usenix.org>

USENIX SUPPORTING MEMBERS

Advanced Resources
ANDATACO
Apunix Computer Services
Auspex Systems, Inc.
Boeing Company
CyberSource Coporation
Digital Equipment Corporation
Earthlink Network, Inc.
Hewlett-Packard
Internet Security Systems, Inc.
Invincible Technologies
Lucent Technologies, Bell Labs
Motorola Global Software
Nimrod AS
Sun Microsystems, Inc.
UUNET Technologies, Inc.
WITSEC, Inc.

of SANE'98,you can register and get the conference materials on the evenings of November 17 and 18. To encourage you to register in the evening, there will a reception on both evenings.

Organization

Program Co-chairs:

Edwin Kremer, *Dept. of Computer Science, Utrecht University*
Jan Christiaan van Winkel, *AT Computing*

Program Committee:

Jos Alsters, *C&CZ, KU Nijmegen*; Bob Eskes, *ASR, Hollandse Signaalapparaten*; Peter den Haan, *C&CZ, KU Nijmegen*; Patrick Schoo, *Dept. of Mathematics, Utrecht University*; Michael Utermöhle, *Dept. of Computer Science, University of Paderborn*; Jos Vos, *X/OS Experts in Open Systems*; Elizabeth Zwicky, *Silicon Graphics, Inc.*

Organization:

Chel van Gennip, *Hiscom;* Mariëlle Klatten, *NLUUG;* Monique Rours, *NLUUG.*

Finally

We will keep you posted and hope you are looking forward to SANE'98. Meanwhile, you can visit us at <http://www.nluug.nl/events/sane98>.

A Second Start for the NLnet Foundation: Short Term Plans

By Teus Hagen, Frances Brazier, Wytze van der Raay, and Jos Alsters

The Board of the NLnet Foundation

To some extent the NLNet Foundation's history (see our article in *;login:*'s April 1998 issue) resembles UUnet's history, at least with respect to its origins in the UNIX Users Group, both in Europe and Holland (NLUUG). NLnet Holding, the company set up by NLnet Foundation, focused on Internet service and backbone provision. At the end of August 1997 the ownership of NLnet Holding was transferred to UUnet. The NLnet Foundation has obtained a position somewhat similar to Advanced Network Services in the US.

The NLnet Foundation now faces new challenges. Its position is unique: the Foundation is financially, commercially, and governmentally independent. Its original goal "to stimulate electronic information exchange" can now be reinstated, with less emphasis on backbone and service provision. The Foundation plans to play its role as a stimulating organization, supporting initiatives for network research, development, and software availability in the public domain.

Beginning May 1, 1998, two of the Executive Board members will be employed full time by the Foundation. All Board members have been active in UNIX User Groups and UNIX networking for more then ten years and are expecting to be able to use their existing connections with UNIX User Groups (in particular NLUUG and USENIX) constructively. This article is one example of this relationship; presence at USENIX meetings (e.g. the annual technical conference in New Orleans) is another.

One of the Internet consortia with which the Foundation is exploring a possible collaboration is the Internet Software Consortium (ISC), with respect to activities such as secure DNS. Collaboration with other Internet consortia is being explored. The goal of the Foundation is to look for loose ends and ideas, to synchronize activities, and to improve distribution channels (e.g., through collaboration with O'Reilly, RedHat, SuSe, etc.). In addition, initiation of new software projects that can contribute to technology improvements and better electronic information exchange are begin pursued.

Collaboration with the Dutch Internet Services Providers Association (NLIP) is another option being examined, in particular with respect to the role the Foundation could play in Holland as a neutral organization for alarm and help desks for children porno information events, CERT type of events, and certification organizational help.

At the end of this year the Foundation will explore possible collaboration with universities and other nonprofit institutions with respect to Internet related research and education. In addition, the Foundation is considering collaboration with a number of software development projects, such as: DNS, INN, DHCP, PGP, ISDN4*NIX, sendmail, IMAP, LDAP, etc. Please do not hesitate to approach the Foundation with new suggestions, keeping in mind that the Foundation's involvement must be more than financial.

Last but not least, here is some information about the people involved. The Foundation has four Board members (general director Teus Hagen, financial director Wytze van der Raay, research director Frances Brazier, director Jos Alsters) and an Advisory Board (Paul de Bra, for network technology, Anne-Marie Kemna (KPMG and the University of Leiden) for legal matters, and Erik Esseling (Moret Ernst and Young) for financial affairs. In May the name NLnet Services will no longer exist as such but will be known as UUNET Nederland BV. From then on the Foundation will appear on the net as nlnet.nl. The Board can be reached via email through the alias <stichting@nlnet.nl> ("Stichting" is the Dutch word for foundation).

The NLnet Foundation will keep you informed about its activities through the Internet, UNIX User Group and USENIX conferences, *;login;* and the NLUUG and other newsletters.

Twenty Years Ago in ;login:

by Peter H. Salus

Peter H. Salus is the author of *A Quarter Century of UNIX* (1994) and *Casting the Net* (1995). He has known Lou Katz for over 40 years.

<peter@pedant.com>

The first UNIX User's Meeting, held at Columbia's College of Physicians and Surgeons on May 15, 1974, drew about two dozen attendees. The meeting held at Columbia University May 24–27, 1978, drew 350. It was not merely the largest user's meeting; it was the most comprehensive.

That range was illustrated in the May and June/July *;login:*s. May featured a long letter from Tom Duff (then at the Computer Grafix Laboratory, New York Institute of Technology), which began:

> By now it is well known that nargs(III) does not work in programmes with separated instruction and data spaces, and that, furthermore, it cannot be made to work by using the mfpi instruction.

Tom then supplied a "version of nargs which . . . with separated I & D space (given a system entry readi(addr) which returns the contents of the given address in instruction space)" Tom's solution worked on 11/45s and 11/70s.

Another bug fix was supplied by Steven Eisen at City University of New York:

> There is a bug in Version 7 of the UNIX C Compiler. The source of the "cc" command is being packaged by Bell Labs with a call to the "alloc" routine of the C Compiler "-lc" library. The new "cc" command, which must be compiled with the "-lS" library, indirectly uses the "malloc" routine of that library. . . . To fix this problem, change the reference in "cc.c" from "alloc" to "malloc" such that "malloc" is used uniformly.

The same *;login:* contained letters from William R. Simmons (Purdue) asking about troff to Versatec interfacing and stating:

> At the present time we are running NEQN and NROFF outputs onto a Diablo 1610 typewriter terminal

and from Baruch Hamel at Vanderbilt. Hamel had two queries: was there a statistical package that would run under UNIX? (They had not been able to compile BMDP or SPSS). "Would you be interested in forming a group of INGRES users within the UNIX users group?"

Since July 1977, this publication has been called *;login:*. Prior to that, it was *UNIX NEWS*. But Mel Ferentz had been phoned by an AT&T lawyer and told that the group (it still had no name) could not use the term UNIX, because they had no permission to do so from Western Electric. At the meeting of UNIX USERS at the College of Physicians and Surgeons, May 24-27, 1978,

> a committee of five people was elected ... with the purpose of proposing a set of bylaws for an organization of users of UNIX* installations. The people elected were:
>
> Mel Ferentz Rockefeller University
> Mars Gralia Johns Hopkins University
> Lou Katz Columbia University Lew Law Harvard University Peter Weiner Interactive Systems Corp.
>
> Law was elected chairman.... The name of the committee shall be the USENIX** committee
>
> * UNIX is a trademark of Bell Laboratories, Inc.
>
> ** USENIX is not a trademark of Bell Laboratories, Inc.

I love the two footnotes: they reveal that the spirit that bonded UNIX users together in the 1970s and continues today had its roots in an "us-against-them" attitude combined with a sense of humor. The two name changes gave rise to a letter to Ferentz from Stephen J. Phillips, a patent attorney at Bell Labs. Phillips' letter began:

> It was with interest that I read *;login:* Vol. 3, No. 6, June/July 1978. Your choice of the name "USENIX" was rather an inspired way of avoiding the use of the UNIX* trademark in the committee name.

Phillips went on at some length about the "dilution of the UNIX mark," which he hoped USENIX "would not incorporate" in any future names. Ferentz, wryly, published the letter in *;login:*.

The name *;login:* is interesting. Dennis Ritchie explained to me:

> The ; was utilitarian. During most of the early '70s the most popular terminal was the Teletype model 37. The sequence <esc>; put it in full-duplex mode so the terminal didn't print characters locally, but let the system echo them. So this sequence was put into the greeting message. Of course it didn't print when you used that terminal, but other terminals that appeared later didn't understand the message and so printed the ; .

Student Proposal Deadlines

The deadlines for submitting proposals for student scholarships, research grants and undergraduate software projects for the 1998–1999 academic year are under reconstruction by the USENIX Scholastic Committee in order to better serve the academic community. New information will be available in early July at <http://www.usenix.org/students/deadlines.html>

Abstracts and status reports of student scholarships, research grants and undergraduate software projects USENIX has funded to date are now available at <http://www.usenix.org/student/sweb/index.html>.

2nd USENIX Windows NT SYMPOSIUM

AUGUST 3-4, 1998
MADISON RENAISSANCE HOTEL, SEATTLE, WASHINGTON
FOR COMPUTER SCIENCE RESEARCHERS USING OR ADOPTING WINDOWS NT AS THEIR BASE

A PREVIEW OF THE FUTURE FROM MICROSOFT CORPORATION

NT FUTURES

Frank Artale
Director, Windows NT Program Management,
Lou Perazzoli
Director, Windows NT Core OS Group,
Felipe Cabrera
Windows NT Architect, Microsoft Corporation

PEER-REFEREED PAPERS SESSIONS

PERFORMANCE

A PERFORMANCE STUDY OF SEQUENTIAL I/O ON WINDOWS NT 4.0
Erik Riedel
Carnegie Mellon University
Catharine van Ingen, Jim Gray
Microsoft Reseach Microsoft Corporation

SCALABILITY OF THE MICROSOFT CLUSTER SERVICE
Werner Vogels, Dan Dumitriu, Ashutosh Agrawal, Teck Chia, Katherine Guo
Cornell University

EVALUATING THE IMPORTANCE OF USER-SPECIFIC PROFILING
Zheng Wang
Harvard University
Norm Rubin
Digital Equipment Corporation

FROM UNIX TO NT TO UNIX

CYGWIN32: A FREE UNIX PORTING LAYER FOR WIN32
Geoffrey J. Noer
Cygnus Solutions

WIN32 API EMULATION ON UNIX FOR SOFTWARE DSM
Sven M. Paas, Thomas Bemmerl, Karsten Scholtyssik
RWTH Aachen

NT-SWIFT: SOFTWARE IMPLEMENTED FAULT TOLERANCE FOR WINDOWS NT
Yennun Huang, P. Emerald Chung, Chandra Kintala
Bell Labs, Lucent Technologies
De-Ron Liang, Chung-Yih Wang
Institute of Information Science, Academia Sinica, Taipei

THREADS

A THREAD PERFORMANCE COMPARISON: WINDOWS NT AND SOLARIS ON A SYMMETRIC MULTIPROCESSOR
Fabian Zabatta, Kevin Ying
Brooklyn College

A SYSTEM FOR STRUCTURED HIGH-PERFORMANCE MULTI-THREADED PROGRAMMING IN WINDOWS NT
John Thornley, K. Mani Chandy, and Hiroshi Ishii
California Institute of Technology

A TRANSPARENT CHECK-POINT FACILITY ON NT
Johny Srouji, Paul Schuster, Maury Bach, Yulik Kuzmin
Intel Corporation

MIXING UNIX AND NT

FILE SYSTEM SECURITY: AN INTEGRATED MODEL FOR NT AND UNIX FILE SERVICE
Dave Hitz, Bridget Allison, Andrea J. Borr, Rob Hawley, Mark Muhlestein
Network Appliance, Inc.

PLUGGABLE AUTHENTICATION MODULES FOR WINDOWS NT
Naomaru Itoi, Peter Honeyman
University of Michigan

MONTAGE—AN ACTIVEX CONTAINER FOR DYNAMIC INTERFACES
Gordon Woodhull, Stephen North
AT&T Laboratories—Research

NETWORKING AND DISTRIBUTED SYSTEMS

SECURESHARE: SAFE UNIX/WINDOWS FILE SHARING THROUGH MULTIPROTOCOL LOCKING
Andrea J. Borr
Network Appliance, Inc.

HARNESSING USER-LEVEL NETWORKING ARCHITECTURES FOR DISTRIBUTED OBJECT COMPUTING OVER HIGH-SPEED NETWORKS
Rajesh Sankaran, Hemal V. Shah
Intel Corporation
Calton Pu
Oregon Graduate Institute of Science and Technology

IMPLEMENTING IPV6 FOR WINDOWS NT
Allison Mankin
University of Southern California,
Richard P. Draves, Brian D. Zill
Microsoft Research Microsoft Corporation

REAL TIME SCHEDULING

A SOFT REAL-TIME SCHEDULING SERVER ON WINDOWS NT
Chih-han Lin, Hao-hua Chu Klara Nahrstedt
University of Illinois

VASSAL: LOADABLE SCHEDULER SUPPORT FOR MULTI-POLICY SCHEDULING
George M. Candea
Massachusettes Institute of Technology
Michael B. Jones
Microsoft Research Microsoft Corporation

SPECIAL SESSIONS

MONDAY EVENING
DEMOS & POSTERS WITH SHERRY RECEPTION

MONDAY EVENING
BIRDS-OF-A-FEATHER SESSIONS

MORE SOCIAL AND NETWORKING ACTIVITIES

SUNDAY EVENING
WELCOME RECEPTION

MONDAY
HOSTED LUNCHEON

TUESDAY EVENING RECEPTION AT THE FLIGHT MUSEUM
Sponsored by Microsoft

Register now. On-line registration: http://www.usenix.org/events/nt98

Large Installation System Administration of Windows NT

LISA NT CONFERENCE

AUGUST 6-8, 1998

MADISON RENAISSANCE HOTEL, SEATTLE, WASHINGTON

FOR MANAGERS OF LARGE INSTALLATIONS OF WINDOWS NT OR HETEROGENEOUS SYSTEMS

Co-sponsored by SAGE, the System Administrators Guild

- Discover workable solutions to the issues of scaling the NT environment
- Learn from the real-life experiences of system administrators who have wrestled with large scale migration from UNIX to NT
- Achieve the high performance of traditional systems with NT
- Hear how others are successfully deploying large applications with NT
- Explore new tools for performance management, security and automated operations
- Tackle the difficulties of connectivity, Internet/ Intranet, security in heterogeneous environments

THE "INSIDE STORY" FROM MICROSOFT CORPORATION

MANAGEMENT DIRECTIONS FOR WINDOWS NT
Frank Artale
Director, Windows NT Program Management

THE MICROSOFT CORPORATE ROLLOUT OF WINDOWS NT 5.0
Sherry Hatfield
Director, Information Technology Group, Enterprise Service and Products

ZERO ADMINISTATION IN WINDOWS NT 5
Dan Plastina
Group Program Manager, Windows NT Server Operating System

PEER-REFEREED PAPERS SESSIONS

MANAGEMENT AND MONITORING

PATCH 32: A SYSTEM FOR AUTOMATED CLIENT OS UPDATES
Gerald Carter
Auburn University

MONITORING USAGE IN AN NT WORKSTATION LAB
Paul Kranenburg
Erasmus University, Rotterdam

JOINTLY MANAGING UNIX AND NT

FILE SYSTEM SECURITY: SECURE NETWORK DATA SHARING FOR NT AND UNIX
Dave Hitz, Bridget Allison, Andrea J. Borr, Rob Hawley, Mark Muhlestein
Network Appliance, Inc.

NT DOMAINS FOR UNIX
Luke Kenneth Casson Leighton

OPERATING SYSTEM AND SOFTWARE INSTALLATION

AUTOINSTALL FOR NT: COMPLETE NT INSTALLATION OVER THE NETWORK
Robert Fulmer
Lucent Technologies, Bell Labs

A COMPARISON OF LARGE SCALE SOFTWARE INSTALLATION ON NT AND UNIX
Michail Gomberg, Remy Evard, Craig Stacey
Argonne National Laboratory

SOFTWARE DISTRIBUTION TO PC CLIENTS IN A COMMON ENTERPRISE NETWORK
Cam Luerkens
Rockwell A&C

ARCHITECTURE AND SERVERS

COMPAQ'S NEW ENGINEERING COMPUTE FARM ENVIRONMENT: MOVING FORWARD WITH WINDOWS NT
Don Brace, Andrew Gordon, Scott Teel
Compaq Computer Corporation

DESIGNING AN OPTIMIZED ENTERPRISE BACKUP SOLUTION WITH INTELLEGENT DATA MANAGEMENT
Kevin Workman, Steven Downs, Mikel Featherstone, Earl Waud
Qualcomm Inc.

PROVIDING RELIABLE NT DESKTOP SERVICES BY AVOIDING NT SERVER
Thomas A. Limoncelli, Robert Fulmer, Thomas Reingold, Alex Levine, Ralph Loura,
Lucent Technologies, Bell Labs

INVITED TALKS FROM THIRD-PARTY EXPERTS

BRINGING THE "REAL" INTERNET TO WINDOWS NT
Bo Ahlberg
MetaInfo, Inc.

TO BE ANNOUNCED
Mark Russinovich
NT Internals

MODERATED PANELS

INTRUSION DETECTION AND HANDLING

WINDOWS NT TIPS AND TRICKS

SPECIAL SESSIONS

WORKS-IN-PROGRESS REPORTS

EVENING BIRDS-OF-A-FEATHER SESSIONS

OLD-FASHIONED NETWORKING

TUESDAY EVENING RECEPTION AT THE FLIGHT MUSEUM
sponsored by Microsoft

THURSDAY AND FRIDAY HOSTED LUNCHEONS

Register now. On-line registration: http://www.usenix.org/events/nt98

USENIX Windows NT TUTORIALS

Madison Renaissance Hotel
Seattle, Washington

All Day Wednesday, August 5th, 1998

Who should attend?

Systems and network administrators of Windows NT systems, as well as architects, developers, and engineers in Windows NT environments

Master complex technologies essential to your success in using, managing and integrating Windows NT

- Take home new skills, comprehensive information, and experts' insight
- Learn hands-on expertise from experienced instructors
- Gain valuable reference guides from printed tutorial materials
- Talk over what you've learned with fellow attendees over hosted luncheons

OUR GUARANTEE: *If you feel a tutorial does not meet the high standards you have come to expect from USENIX, let us know by the first break and we will change you to any available tutorial immediately.*

W1 *(full-day)* WINDOWS NT SECURITY

Rik Farrow
Consultant

Master the complex security model of Windows NT and discover how to maximize the security of networked NT systems. Learn how to correctly configure NT's security features, which can satisfy the most avid control freak. On the other hand, find out how to secure NT's API, which has been a primary target for virus writers, and how to handle Windows applications written with little security. Those accustomed to UNIX systems will benefit from comparisons between UNIX and NT.

TAKE AWAY AUTHORITATIVE, IN-DEPTH KNOWLEDGE OF:

- Command line interfaces and tools for controlling and auditing as well as NT's security-related GUIs
- The NT registry and associated access control lists
- User accounts, local and global groups, rights, and privileges
- Domains, domain controllers, local and network authentication
- NT Passwords, and collecting and cracking passwords
- ACLs for file, directories, and other objects
- Event and audit mechanism
- IIS, RAS, network services and firewall systems for NT

Rik Farrow *provides security consulting and teaches throughout North America and Europe. He is the author of* UNIX System Security *(Addison-Wesley) and* System Administrator's Guide to System V *(Prentice Hall). Farrow is a columnist for* ;login: *and for* Network Magazine.

W2 *(full-day)* WINDOWS NT INTERNALS

Jamie Hanrahan
Kernel Mode Systems

Get the most from Windows NT, both versions 4 and 5, by understanding its underlying architecture and operation from the Windows applications developer's point-of-view. This in-depth look into the general system architecture is a big help for system administrators too. And, since "Designed for Windows 95" doesn't guarantee "runs on Windows NT" this tutorial will help you ensure reliable application compatibility.

TAKE FULL ADVANTAGE OF SUCH NT FEATURES AS:

- Asynchronous I/O, preemptive multitasking, and multiprocessor support
- Calling system functions from applications
- Environment subsystems
- Thread scheduling
- Memory management internals
- Using and interpreting performance measurement tools
- Windows 9x & Windows NT applications compatibility checklist

Jamie Hanrahan *provides Windows NT driver development, consulting, and training services to leading companies. He received the Instructor of the Year award while teaching courses for Digital Equipment Corporation. He is co-author of* VMS Advanced Driver Techniques *(O'Reilly and Associates) and is now co-writing a book on Windows NT device drivers.*

For detailed tutorial descriptions, please go to http://www.usenix.org/events/nt98

W3AM *(morning half-day)*
Configuring Samba: Avoiding Pitfalls

John Blair
University of Alabama, Birmingham

You can provide Windows (SMB) networking services conveniently and inexpensively in your heterogeneous environment with the popular and freely-distributed Samba software suite. Samba allows Windows 95, Windows NT, and OS/2 machines to "see" hosts running UNIX and UNIX-like operating systems on their network, including the Internet. But the issues Samba tackles are complex, so configuring it can be difficult. You will learn how to avoid the simple to complex configuration problems.

Go home with a thorough understanding of Samba and its components, and a toolbox of solutions to:

- Deal with problems caused by the differences between the UNIX and Windows filesystems
- Use Samba to process logins from Windows NT and Windows 95 machines
- Smoothly integrate a Samba server into a large Windows network containing multiple NT domains and spanning more than one TCP/IP subnet

John Blair *is the author of* Samba: Integrating UNIX and Windows *(SSC) and a frequent contributor to* Linux Journal. *As a member of the Samba Team, he has contributed documentation improvements and minor bits of code.*

W4AM *(morning half-day)*
Windows NT Performance Monitoring and Tuning

Aeleen Frisch
Exponential Consulting

Find out in-depth how to monitor and then improve the performance of your system and network running on Windows NT.

System administrators (who are familiar with elementary Windows NT concepts and tasks) will gain mastery of the standard Windows NT Performance Monitor facility. A variety of real-world scenarios and examples enhance the discussion.

Learn all about:

- Overview of performance factors and considerations
- Monitoring system operation and evaluating system efficiency
- Strategies for locating performance bottlenecks
- CPU performance
- Memory usage and its performance implications
- Disk and other I/O performance issues
- Network performance issues
- System tuning strategies and hints
- Capacity planning

Aeleen Frisch *has been a system administrator for over 15 years. She currently looks after a very heterogeneous network of UNIX and Windows NT systems. She is the author of several books, including* Essential Windows NT System Administration *(O'Reilly and Associates).*

W5PM *(afternoon half-day)*
Understanding COM and MTS

David Chappell
Chappell & Associates

For architects and developers who need to understand the basics of COM and MTS. Knowledge of a specific programming language, such as C++, is not required.
Get a thorough grounding in COM and MTS. These key Microsoft technologies are essential to architects and developers working in Windows NT. The Component Object Model (COM) is fundamental to Windows and Windows NT software development. The COM-based Microsoft Transaction Server (MTS) supports the creation of scalable, transaction-oriented servers by combining the relatively new notion of component software with familiar concepts from transaction processing monitors.

In this introductory tutorial, you'll learn about:

- COM definitions: Interfaces, classes, life cycle, and more
- How MTS adds automatic support for transactions to COM
- How MTS adds scalability to COM servers
- Comparing MTS components and Enterprise Java Beans

David Chappell *is Principal of Chappell & Associates, an education and consulting firm. He presents seminars around the world and writes a regular column for* Object Magazine. *His most recent book is* Understanding ActiveX and OLE *(Microsoft Press).*

W6PM *(afternoon half-day)*
DHCP & DNS

Greg A. Kulosa
GNAC, Inc.

Do you want to distribute information automatically to networked Windows clients without needing to manually configure each machine? Learn all about how to use DHCP to distribute IP address, router, DNS, WINS and other network information for Windows NT clients, with pointers to how UNIX clients are similar. Next, master DNS the — Domain Name Service — by which Internet TCP/IP hosts look up host addresses and network services. We will thoroughly cover the DHCP and DNS protocols, and how these protocols fit into a typical network with both UNIX and Windows NT servers.

Manage your networks more efficiently with DHCP & DNS:

- DHCP & DNS protocols in-depth
- Which server platform should be used? UNIX or Windows NT?
- Which DHCP & DNS server should be run? Should you use a commercial solution or will freeware do the job?
- Daily maintenance of DHCP and DNS servers
- How to integrate DHCP information into DNS (And do you really need to?)
- Debugging problems when they occur
- Useful reference materials

Greg A. Kulosa *is an experienced systems administrator. He has rolled out numerous mutiplatform DHCP networks.*

Attend these tutorials and get an immediate payoff: command of essential technologies you can put right to work at your site.

Call for Papers

USENIX

3rd Symposium on Operating Systems Design and Implementation (OSDI '99)

Sponsored by USENIX and Co-sponsored by ACM SIGOPS and IEEE TCOS

The complete Call for Papers can be found at http://www.usenix.org/events/osdi99

February 22–25, 1999 **New Orleans, Louisiana, USA**

Important Dates

Full papers due: ***July 28, 1998***
Notification to authors: *October 13, 1998*
Revised papers due for shepherding: *December 1, 1998*
Camera-ready full papers due: *January 6, 1999*

Continuing in the tradition of the OSDI symposia, the third OSDI will continue to focus on practical issues related to modern operating systems. OSDI brings together professionals from academic and industrial backgrounds and has become the perfect forum for issues concerning the design and implementation of operating systems for modern computing platforms such as workstations, parallel architectures, mobile computers, and high speed networks.

The OSDI symposium emphasizes both innovative research and quantified experience in operating systems. We seek contributions from all fields of operating systems practice: new ideas, the influence of new hardware and networking on systems, evaluations of existing systems, etc. The stress is on practice: What can we learn from trying to build systems that work well? Sometimes they don't work well; OSDI encourages the submission of papers that discuss the reasons for systems' failures as well as successes.

How to Submit a Paper to the Refereed Track

Authors are required to submit full papers by July 28, 1998. Submitted papers should be no longer than 14 single spaced 8.5" x 11" pages, including figures, tables, and references. Papers longer than 14 pages will not be reviewed. Papers so short as to be considered "extended abstracts" will not receive full consideration.

A good paper will demonstrate that the authors:

- are attacking a significant problem
- are familiar with the literature
- have devised an original or clever solution
- have implemented the solution and characterized its performance
- have drawn appropriate conclusions

The papers will be judged on interest, significance, originality, clarity, relevance, and correctness. The committee will favor papers with reproducible results, especially those supplying detailed data and explanations, or offering to make data sets or source code available. Accepted papers will be shepherded through an editorial review process by a member of the program committee.

Note: OSDI, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors of accepted papers must provide an HTML page containing the abstract and links to their paper, slides, and software, if available. This will be collected after the event for inclusion in an electronic version of the symposium. For examples, see

www.cs.utah.edu/~lepreau/osdi94/ and
www.usenix.org/publications/library/proceedings/osdi96/

Specific questions about submissions may be sent to the program chairs via email to *osdi99chairs@usenix.org.*

Where to Submit

Submission of all papers must be made in both paper and electronic form. Fifteen (15) paper copies (double-sided if possible) of the paper must be sent to:

Margo Seltzer
Division of Engineering and Applied Science
Harvard University
28 Oxford St.
Cambridge, MA 02138
617.496.5663

and one electronic copy in PostScript (not ASCII) must be submitted by electronic mail to *osdi99papers@usenix.org*

For administrative reasons (not blind reviewing), every submission (in both its paper and electronic form) should include one additional page containing (i) paper title and authors, indicating any who are full-time students, and (ii) for the author who will act as the contact to the program committee, his or her name, paper mail address, daytime and evening phone numbers, email address and fax number, if available. The cover sheet mailed with the electronic paper submission should be in ASCII to facilitate accurate on-line bookkeeping and should be included in the same electronic mail message as the PostScript file containing the paper.

For more details on the submission process, authors are encouraged to consult

www.usenix.org/events/osdi99/guidelines.html

or send mail to *osdi99authors@usenix.org.*

All submissions will be acknowledged by July 31, 1998. If your submission is not acknowledged by this date, please contact the program chairs promptly at *osdi99chairs@usenix.org.*

EurOpen.se 1999 USENIX

Announcement and Call for Papers

NordU99 - The First Nordic EurOpen/USENIX Conference February 9-12, 1999 Grand Hotel, Stockholm, Sweden

A conference organized by EurOpen.SE – The Swedish Association of Unix Users, and an affiliate of USENIX, the Advanced Computing Systems Association

Important Dates:

Extended abstracts due: August 14, 1998
Notification of acceptance: September 12, 1998
Final papers due: November 28, 1998

Authors are invited to submit a 1-page (maximum) abstract in English on any of the topics below to the Congress Secretariat. Submissions should be original work and will be reviewed by the Technical Programme Committee. All accepted papers will be available on a CD–ROM which will be given to all attendees. Authors must register for the conference and present their papers in person. Instructions for preparing final papers will be sent with the letter of acceptance.

Complete program and registration information will be available by mid September 1998. Topics:

Security tools and techniques
Electronic Commerce
Electronic Publishing
Innovative system adminstration tools & techniques
Performance analysis, monitoring and tuning
Networking

Please send your abstracts to:

Congrex Sweden AB
Attn: EurOpen '99
P.O. Box 5619
SE–114 86 Stockholm
SWEDEN

Phone: +46 8 459 66 00
Fax: +46 8 6619125
E–mail: congrex@congrex.se

Conference URL

http://www.europen.se/NordU99/

Announcement and Call for Papers USENIX

2nd Conference on Domain-Specific Languages

Sponsored by USENIX and co-sponsored by ACM SIGPLAN in cooperation with ACM SIGSOFT

October 3-6, 1999
Omni Hotel
Austin, Texas

Conference URL: *http://www.usenix.org/events/dsl99/*

Important Dates for Refereed Papers

Papers due: *March 22, 1999*
Author notification: *June 2, 1999*
Camera-ready final papers due: *August 24, 1999*

Conference Organizers

Program Chair
Thomas Ball, *Lucent Technologies*
Program Committee
Tim Bray, *Textuality*
Charles Consel, *University of Rennes / Irisa*
Mary Fernandez, *AT&T Labs Research*
Paul Hudak, *Yale University*
James Larus, *University of Wisconsin - Madison*
Doug Lea, *State University of New York at Oswego*
Jay Lepreau, *University of Utah*
Brad Myers, *Carnegie Mellon University*
Todd Proebsting, *Microsoft Research*
David S. Rosenblum, *University of California, Irvine*
Michael Schwartzbach, *University of Aarhus*
Invited Talks Coordinator
Carlos Puchol, *Lucent Technologies*

Introduction

Language is central to the discipline of software engineering. Programmers use a variety of languages in their daily work, and new languages appear frequently. These languages may offer new solutions to problems such as software production, distribution, maintenance and enhancement. However, not all languages address the problem of general-purpose computing: domain-specific languages (DSLs) are explicitly designed to address a particular class of problems, while offering compelling advantages within that class.

This conference is dedicated to the discussion of the unique aspects of DSL design, DSL implementation, and the use of DSLs in software engineering.

Domain-specific languages give rise to a number of questions:

What are the design principles for the creation of new DSLs?

What are the concrete technical advantages and disadvantages of DSLs?

What roles can DSLs play in software engineering, and how does their use affect software engineering process?

What are the tools, environments, and techniques needed to support DSLs?

What are the economic costs and benefits of DSLs?

These and other questions are the focus of this conference, which seeks to advance the practice of DSL design, DSL implementation, and software engineering generally by:

- eliciting examples of successful DSLs
- highlighting the spectrum of benefits which DSLs can provide (e.g., compile-time guarantees of behavior, improve program performance, reduced interval, ...)
- discovering design principles and methodologies for creating DSLs
- eliciting design techniques and tools for working with DSLs throughout the software engineering lifecycle
- providing a framework within which language designers from different domains can easily communicate
- establishing the practical value of DSLs through the publication of empirical data concerning productivity, quality, and maintainability
- creating a community that will continue to study and refine the practice of software engineering through DSLs

Conference Topics

The technical sessions will include refereed papers, invited talks, tutorials, and Birds-of-a-Feather (BoF) sessions. We seek papers that draw on experience in a wide variety of areas, including but not limited to the following topics:

Example Domains:

Arts
- graphic arts
- architecture (CAD/CAM)

Sciences
- computational biology/chemistry
- DNA sequencing
- medical instrument control
- image analysis

Computers
- databases/heterogenous data management
- graphics/multimedia
- extensible operating systems
- structured documents (SGML, XML)
- active networks
- distributed and parallel computing

Entertainment
- games
- animation

Music
- composition

Computer Science:
- declarative languages
- object-oriented languages
- visual languages and environments
- software design, specification, and architecture
- software engineering and software process
- language implementation infrastructure
- program analysis and automated transformation
- formal methods
- type theory
- testing/prototyping

Best Paper Awards

Awards will be given for the best paper and best student paper at theconference.

Paper Criteria

Authors are required to submit full papers by March 22, 1999. Submitted papers should be no longer than 14 single spaced 8.5" x 11" pages, including figures, tables, and references. Papers longer than 14 pages will not be reviewed.

Papers will be judged on the depth of their insight and the extent to which they translate specific experience into general lessons for domain-specific language designers, implementers, and software engineers.

Papers can range from the practical to the theoretical; papers should refer to actual languages, tools, and techniques with pointers to full definitions and implementations where possible. Empirical data on results should be included where possible. Case studies are also welcome. A good paper will demonstrate that the authors:

- are attacking a significant problem
- are familiar with the literature
- have devised an original or clever solution
- have implemented the solution and characterized its performance
- have drawn appropriate conclusions

For detailed author guidelines, send email to **dsl99authors@usenix.org**.

Note: DSL, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors of accepted papers will be expected to provide both a PostScript file of the final paper and an HTML version that includes links to their slides and software, if available. These will be included in an electronic version of the conference (see http://www.usenix.org/publications/library/proceedings/dsl97)

Specific questions about submissions may be sent to the program chair via email to **dsl99chair@usenix.org**.

Publication

The conference papers will be published in the conference proceedings and in an issue of SIGPLAN *Notices,* the primary informal monthly publication of the Special Interest Group on Programming Languages (SIGPLAN) of ACM.

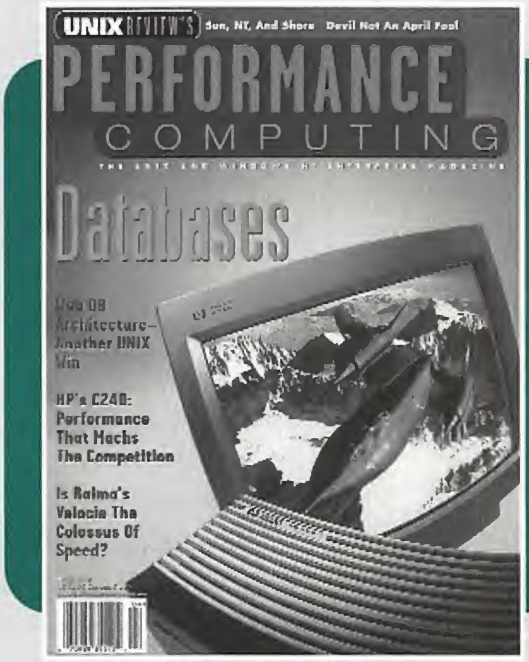
How/Where to Submit

Submission of all papers must be made in both paper and electronic form. Fifteen (15) paper copies (double-sided if possible) of the paper must be sent to:

Thomas Ball
Room 2A-314
Bell Laboratories, Lucent Technologies
263 Shuman Boulevard
Naperville, IL 60566-7050

One electronic copy in PostScript (not ASCII) must be submitted by electronic mail to **dsl99papers@usenix.org**, accompanied by the author information form at **http://www.usenix.org/events/dsl99/authorinfo.txt** (MIME attachments are allowed).

The Only Magazine Dedicated to Technologists Building, Managing, and Integrating UNIX and Windows NT in an Enterprise-Computing Environment



Free Subscription

Apply at *www.performancecomputing.com*
The High-Performance Web Site
For The Enterprise Technologist.

UNIX REVIEW'S
PERFORMANCE
COMPUTING

Local User Groups

UNIX and LINUX Groups

The USENIX Association will support local user groups by doing a mailing to assist in the formation of a new group and publishing information on local groups in *:login;* and on its Web site. Full details can be found at: <http://www.usenix.org/membership/LUGS.html>.

At least one member of the group must be a current member of the Association.

Send additions and corrections to: <login@usenix.org>

California

Bay Area

Bay Area FreeBSD User Group

San Francisco Bay Area Linux Users' Group

Orange County

UNIX Users Association of Southern California (UUASC)

Silicon Valley

Silicon Valley Linux Users' Group

Colorado

Boulder

Boulder Linux Users Group

Front Range UNIX Users Group

Connecticut

The Connecticut Free UNIX Group

District of Columbia

Washington

Washington Area UNIX Users Group

Florida

Orlando

Central Florida UNIX Users Group

Western

Florida West Coast UNIX Users Group

Kansas/Missouri

Kansas City

Kansas City UNIX Users Group (KCUUG)

Massachusetts

Worcester

WPI Linux Association

Worcester Linux User's Group

Michigan

Detroit/Ann Arbor

Southeastern Michigan Sun Local Users and Nameless UNIX Users Group

Missouri

St. Louis

St. Louis UNIX Users Group

New England

Northern New England UNIX Users Group (NNEUUG)

New Mexico

Albuquerque

ASIGUNIX

New York

New York City

Unigroup of New York City

Oklahoma

Tulsa

Tulsa UNIX Users Group, $USR

Tennessee

Nashville

Nashville Linux User Group

Texas

Austin

Capital Area Central Texas UNIX Users Society (CACTUS)

Dallas/Fort Worth

Dallas/Fort Worth UNIX Users Group

Houston

Houston UNIX Users Group (HOUNIX)

Washington

Seattle

Seattle UNIX Group

Armenia

Yerevan

The Armenian UNIX Users Group (AMUUG) was founded in December 1996. AMUUG is open to all interested individuals and organizations, regardless of affliation, without any fee.

Canada

Alberta

Calgary UNIX Users Society (CUUG)

Manitoba

Manitoba UNIX User Group (MUUG)

Ontario

Toronto Group

Ottawa Carleton UNIX Users Group (OCUUG)

Ottawa Carleton Linux Users Group (OCLUG)

System Administration Groups

SAGE supports local groups. Full listing of group Web sites and other details can be found at:

http://www.usenix.org/sage/locals/

ASUQ
Meets first Wednesday of every third month in Montreal, Quebec.

AZSAGE
Meets monthly in the Phoenix area.

BayLISA
Serves the San Francisco Bay Area.

Back Bay LISA (BBLISA)
Serves the Boston, Massachusetts and New England area.

Beach LISA
A group for system administrators in the San Diego area hosts monthly meetings and the occasional social event.

CintiSAGE
CintiSAGE, a group for the Greater Cincinnati/ Northern Kentucky area, is now forming. To get on the mailing list, send mail with 'subscribe' in the body to cinti-sage-request@ebb.org.

dc.sage
Serves the Washington D.C. Area.

Dallas-Fort Worth SAGE (dfwsage)
Serves the North Texas area.

$GROUPNAME
Serves the New Jersey area.

EnglishBayLISA
Serves the Vancouver, British Columbia and the British Columbia lower mainland.

Houston Area Sysadmins (HASH)
Serves the greater Houston Area. Join the mailing list by sending a subscribe message to hash-request @tree.egr.uh.edu

Los Angeles Area Group
A group in Los Angeles is being formed. Please contact Josh Geller (joshua@cae.retix.com) if you are interested.

New York Systems Administrators (NYSA)
Serves the New York City area.

North Carolina System Administrators
Serving central North Carolina, particularly the Triangle Area.

Old Bay SAGE
A group for sysadmins in the greater Baltimore Maryland area.

SAGE-AU
The System Administrators Guild of Australia

Seattle SAGE Group (SSG)
Seattle, Washington area.

Twin Cities Systems Administrators (TCSA)
Serving the Twin Cities and surrounding areas of Minnesota.

motd

by Rob Kolstad

Rob was first certified as an open water diver in 1977. He dives around the world, usually on a dive boat that caters to his style: sleep, eat, and dive. He's contemplating diving the Galapagos or Solmar V (manta rays off Baja California) for his next major trip.

<kolstad@usenix.org>

Renewals and Beginnings

Every so often, I write about having taken a vacation. This is another one of those columns. Isn't it unfortunate that the last one was two and a half years ago?

Vacations come in so many flavors, of course. There's the "long weekend" with a quick trip down to Monterey (for West Coasters) or somesuch. There's the "mini-vacation" — Saturday through Tuesday at Disney World with the kids or a quick fishing trip in the mountains. I think these all would be cool if taken on a frequent basis. Regrettably, I'm never that smart.

The week-long vacation (Saturday through Saturday, for example) is the minimum size these days to get me mentally "away from the office." I spent a week on a dive boat last week and was truly away — not a single phone call or e-mail message. I am informed that the new replacement boat (due within a month) offers both of these services to guests!

The two-week or other marathon vacation is a bird of myth. I've known people to sight it, but I am not one of them.

My single week away sure has relaxed my thinking so that I can see the world through somewhat rosier glasses.

The U.S. Open programming contest just finished, so we're busy picking the finalists for the programming camp. It's good to see the old hands finishing so well and the new ones coming up to challenge them. The veterans (some of whom have two or three camps under their belt at age 16 or 17) are being battered by some of the young whippersnapper newcomers. Very refreshing.

The office has a brighter atmosphere than I remembered. It's great to talk to people on the phone and take the extra 30 seconds to be sociable and discuss any random issue of the day. Sure makes things more pleasant.

People are much more pleasant since I've been gone! I am not even mad at my racketball doubles partner who ditched me one night for a "better" partner . . . or even at the guy who smashed the ball directly into my nose. I was having trouble forgiving him for that one, since I had "called" the shot as mine.

My house, unchanged since my departure, looks tidier somehow. The sunlight filters more happily into it.

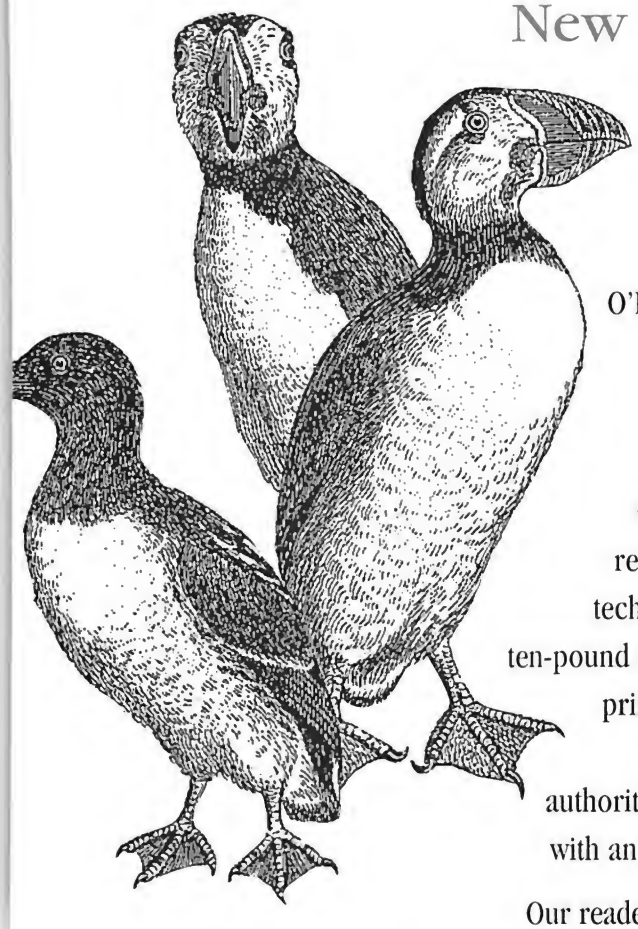
Viewing my dive pictures almost warms my heart. Ten years into this particularly vexing hobby, I'm finally taking fewer pictures that lack that certain "something" (like focus, light, or a subject). They're now colorful and entertaining. Some of them are even suitable for showing friends. Yay.

I can't understand how my absence causes all these things, but the correlation is clear to me. I'm wondering if I should get away more often.

If you're viewing the world through ever darker glasses, finding your coworkers or friends (or relatives or spouses or children) being ever less reasonable in dealing with you, or losing your patience in daily affairs, you might consider doing what I did. Go somewhere without them for a week. Some chemistry happens that just seems to make things right.

Bon voyage!

A Different Kind of Animal



New UNIX and System Administration Books from O'Reilly

O'Reilly creates books for professionals like you — people with serious information needs who don't want run-of-the-mill answers.

Our readers depend on us to provide reliable, no-nonsense solutions to their technical problems. You'll never find any ten-pound doorstops that have been rushed into print here. Instead, you can count on us to produce meticulously researched, authoritative books written by the experts — with an unbiased, independent point of view.

Our readers know that. That's why they trust us.

Managing Usenet
By Henry Spencer & David Lawrence
1st Edition 1/98, 508 pages
ISBN 1-56592-198-4, $32.95

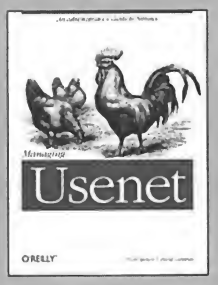
Managing Mailing Lists
By Alan Schwartz, 1st Edition 3/98
298 pages, ISBN 1-56592-259-X, $29.95

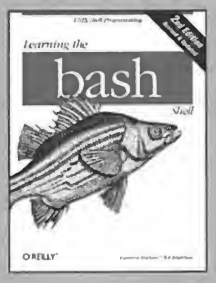
Learning the bash Shell, 2nd Edition
By Cameron Newham & Bill Rosenblatt
2nd Edition 1/98, 336 pages
ISBN 1-56592-347-2, $29.95

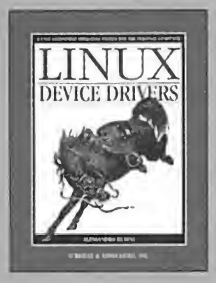
Linux Device Drivers
By Alessandro Rubini, 1st Edition 2/98
442 pages, ISBN 1-56592-292-1, $29.95

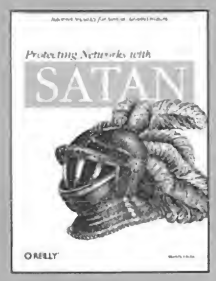
Protecting Networks with SATAN
By Martin Freiss, 1st Edition 6/98
152 pages, ISBN 1-56592-425-8, $19.95

Virtual Private Networks
By Charlie Scott, Paul Wolfe & Mike Erwin
1st Edition 3/98, 192 pages
ISBN 1-56592-319-7, $29.95









O'REILLY™
www.oreilly.com

101 MORRIS STREET, SEBASTOPOL, CA 95472

ORDERS/INQUIRIES: **800-998-9938** WEEKDAYS 6AM-5PM PST

707-829-0515 • FAX: 707-829-0104

EMAIL FOR OUR CATALOG: **CATALOG@OREILLY.COM**
INCLUDE YOUR NAME AND MAILING ADDRESS

O'REILLY BOOKS ARE ALSO AVAILABLE AT YOUR BOOKSTORE

Please mention code
ALOG
when ordering to receive your **20%** USENIX discount

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

AUTOMATIC INFORMATION SERVER

If you do not have access to the Web, finger <info@usenix.org> and you will be directed to the catalog which outlines all conferences, activities, and services.

PGP INFORMATION

All correspondence to:
1998 Operational Key
Key ID: 1024/F6F82613 1997/11/21
USENIX 1998 <pgp@usenix.org>
Key fingerprint = 80 6F B5 48 C2 1A B8 45 48 5F F2 38 E6 41 B0 61

1998 Signing Key
Key ID: 1024/4F75A901 1997/11/21
USENIX 1998 Signature
<http://www.usenix.org/pgp/pgpsig.html>
Key fingerprint = 05 FD CF A1 2F 47 00 5C 69 C2 25 E4 66 89 A6 9B

USENIX master key <not-for-email>:
Key ID: 1024/2FEA2EF1 1996/04/08
Key fingerprint = DB A7 50 99 66 E4 8A A9 80 B2 D9 E2 FE DA 00 5E

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are August 11, 1998 and October 6, 1998.

ADVERTISING ACCEPTED

;login: offers an exceptional opportunity to reach 9,000 leading technical professionals worldwide.

Contact: Cynthia Deno
cynthia@usenix.org
408 335 9445

USENIX

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAG
PAI
AT BERKELEY, CALIFORNI
AND ADDITIONAL OFFICE